TUGAS AKHIR - KS141501

PENGEMBANGAN APLIKASI SISTEM PEMANTAU ARMADA UNTUK DISTRIBUSI BARANG BERBASIS ANDROID

DEVELOPMENT OF FLEET MONITORING SYSTEM FOR GOODS DISTRIBUTION APPLICATION BASED ON ANDROID

JOHANNES DUMOLI TAMBUNAN
NRP 5213 100 139

Dosen Pembimbing
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.
NIP. 197302191998021001

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR - KS141501

PENGEMBANGAN APLIKASI SISTEM PEMANTAU ARMADA UNTUK DISTRIBUSI BARANG BERBASIS ANDROID

JOHANNES DUMOLI TAMBUNAN
NRP 5213 100 139

Dosen Pembimbing
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.
NIP. 197302191998021001

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT - KS141501

DEVELOPMENT OF FLEET MONITORING SYSTEM FOR GOODS DISTRIBUTION APPLICATION BASED ON ANDROID

JOHANNES DUMOLI TAMBUNAN
NRP 5213 100 139

Supervisor
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.
NIP. 197302191998021001

DEPARTMENT OF INFORMATION SYSTEM
Faculty Of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

LEMBAR PENGESAHAN

PENGEMBANGAN APLIKASI SISTEM PEMANTAU ARMADA UNTUK DISTRIBUSI BARANG BERBASIS ANDROID

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen dan Keamanan Teknologi Informasi
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

JOHANNES PUMOLI TAMPINJAN
NRP 5213 100 139

Surabaya, 29 Januari 2018

PLH KEPALA
DEPARTEMEN SISTEM INFORMASI

Edwin Riksakomara, S.Kom., M.T.
NIP 19690725200312100 1

LEMBAR PERSETUJUAN

PENGEMBANGAN APLIKASI SISTEM PEMANTAU ARMADA UNTUK DISTRIBUSI BARANG BERBASIS ANDROID

TUGAS AKHIR
Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Infrastuktur dan Keamanan Teknologi Informasi
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Insitut Teknologi Sepuluh Nopember

Oleh:

JOHANNES DUMOLI TAMBUNAN
NRP 5213 100 139

Disetujui Tim Penguji: Tanggal Ujian: 10 Januari 2017
Periode Wisuda: Maret 2018

Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom. **(Pembimbing I)**

Bekti Cahyo Hidayanto, S.Si., M.Kom. **(Penguji I)**

Nisfu Asrul Sani, S.Kom., M.Sc. **(Penguji II)**

PENGEMBANGAN APLIKASI SISTEM PEMANTAU ARMADA UNTUK DISTRIBUSI BARANG BERBASIS *ANDROID*

Nama Mahasiswa : Johannes Dumoli Tambunan
NRP : 05211340000139
Jurusan : Sistem Informasi FTIf – ITS
Pembimbing 1 : Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.

ABSTRAK

Sebuah perusahaan penyedia kebutuhan fisik manusia membutuhkan armada kendaraan dalam menyalurkan produk atau layanannya dengan cepat. Namun saat ini, perusahaan masih kesulitan dalam memonitor kinerja armadanya, apakah sedang bekerja, apakah mengantarkan barang pada jalur yang seharusnya, dan apakah barang yang sampai sesuai dengan yang dibawa dari gudang. Sudah ada beberapa perusahaan yang mengembangkan aplikasi yang bisa memonitor armada, namun masih sedikit dan tergolong mahal. Untuk membantu memenuhi kebutuhan dalam memantau armada perusahaan, penelitian tugas akhir ini akan membuat sebuah aplikasi yang berbasis Android agar perusahaan dapat melacak armadanya sesuai dengan permasalahan diatas. Pemilihan aplikasi yang berbasis Android ini sudah berdasarkan perti mbangan harga perangkat yang murah, perangkat sudah digunakan oleh banyak lapisan masyarakat baik yang berekonomi rendah maupun tinggi, pendistribusian aplikasi di Android yang gratis, dan tidak memerlukan alat khusus yang ditambahkan dalam kendaraan. Perancangan desain dan prototype aplikasi dalam penelitian ini menggunakan Google Design Sprint yang dikembangkan oleh Google Venture mengingat beberapa batasan, seperti waktu penelitian yang cukup singkat, belum ada gambaran yang jelas bagaimana produk akhir akan seperti apa, memungkinkan dalam adanya perubahan scope proyek, dan perubahan standar produk yang akan dibuat. Dalam

metode pengerjaannya design sprint hanya membutuhkan waktu 5 hari sampai desain dan validasinya selesai. Lalu dalam pengembangan database aplikasi, penelitian ini menggunakan database yang disediakan oleh Firebase yakni Realtime firebase sehingga database aplikasi dapat tersimpan pada cloud dan masih gratis digunakan oleh 100 user. Aplikasi pemantau armada terdiri dari 3 aplikasi yakni aplikasi berbasis Android untuk manajemen perusahaan yang bernama Fleemo Company, aplikasi berbasis Android untuk sopir armada yang bernama Fleemo Driver, dan aplikasi berbasis web untuk manajemen perusahaan yakni Fleemo Website. Dengan Fleemo Company, perusahaan dapat memberikan tugas pada sopir dengan mudah tanpa harus bertemu langsung, dapat memantau lokasi armada secara realtime, memantau riwayat lokasi armada, dan memantau keberhasilan sopir menyelesaikan tugasnya. Dengan Fleemo Driver, sopir dapat menerima tugas secara realtime, memberikan kordinat lokasinya terkini, dan memberitahukan perusahaan penyelesaian tiap tugas yang diberikan. Dan Fleemo Website dapat mempermudah perusahaan melihat lokasi-lokasi armada dalam tampilan layar besar.

Kata Kunci: Android, Design Sprint, Firebase, Fleet, Google maps, Monitoring

DEVELOPMENT OF *FLEET MONITORING SYSTEM* FOR GOODS DISTRIBUTION APPLICATION BASED ON *ANDROID*

Student Name : Johannes Dumoli Tambunan
NRP : 05211340000139
Department : Sistem Informasi FTIf – ITS
Supervisor 1 : Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.

ABSTRACT

A provider of physical needs of human beings in need of fleet vehicle in channeling their products or services quickly. But this time, the company was still the difficulty in monitoring the performance of the fleet, is currently working, would deliver the goods on the line should be, and whether the goods are match with what was taken from the warehouse. There are already some companies that develop applications that can monitor the fleet, but still a bit and is expensive. So to help meet the needs in the company's fleet monitoring, this final project research will create an Android-based applications so that the company can track the fleet in accordance with the problems above. The selection of Android-based applications is already based on consideration of the cheap device price, the device is already in use by many layers of society either low or high, the economical distribution of applications in the Android is free, and does not require special tools added in the vehicle. Design design and prototype applications in this study using Google Sprint Design developed by Google Venture considering some restrictions, such as time of research were quite short, there hasn't been a clear description of how the product the end will be like what, allow any change in the scope of the project, and changes to the

standard of the product to be created. In the method of work design sprint only takes 5 days to design and been done. Then in the development of database applications, this research uses of databases provided by the Firebase namely Realtime firebase databases application can be stored in the cloud and is still free to use by 100 users. The application monitors the fleet consists of 3 applications Android-based applications for the management of the company by the name of Fleemo Company, Android-based applications for a driver named Fleemo fleet drivers, and web-based applications for management the company i.e. Fleemo Website. With the Fleemo Company, the company can provide a task to driver easily without having to meet in person, can monitor in realtime, monitor fleet location history, and monitor the achievement of driver in completing his task. With the Fleemo Driver, the driver can receive tasks in realtime, giving the coordinates of the current location, and notify the company the completion of each task provided. And the Fleemo Website can make company look at fleet locations in the large screen display.

Keywords: Android, Design Sprint, Firebase, Fleet, Google maps, Monitoring

KATA PENGANTAR

Segala puji dan syukur dipanjatkan kehadirat Tuhan Yang Maha Esa atas segala petunjuk, pertolongan, dan kekuatan yang diberikan pada seluruh umatNya. Hanya atas rahmat dari Tuhan, peneliti dapat menyelesaikan Laporan Tugas Akhir dengan judul:

PENGEMBANGAN APLIKASI SISTEM PEMANTAU ARMADA UNTUK DISTRIBUSI BARANG BERBASIS *ANDROID*

yang merupakan syarat terakhir kelulusan dalam rangka mendapatkan gelar Sarjana Komputer pada Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Insitut Teknologi Sepuluh Nopember Surabaya.

Peneliti mengucapkan terima kasih yang sebesar-besarnya pada semua pihak yang telah membantu dalam menyelesaikan Tugas Akhir ini, terkhusus pada:

1. Tuhan Yang Maha Esa yang telah memberikan kesempatan, kepintaran, kekuatan, kesehatan, kasih saying, petunjuk, dan waktu yang cukup dalam penyelesaian Tugas Akhir ini.
2. Bapak Robert Tambunan dan Yanti Hutahaean selaku orang tua peneliti yang tiada henti memberikan dukungan doa, semangat, mengingatkan arti berjuang dan bersyukur, tidak lupa akan Tuhan dan segala bentuk dukungan materi. Terima kasih, kerja keras kalian tidaklah sia-sia.
3. Bapak Dr. Ir. Aris Tjahyanto, M.Kom. selaku Ketua Jurusan Sistem Informasi.

4. Bapak Rully Agus Hendrawan, S.Kom., M. Eng. Selaku dosen wali yang telah memberikan banyak pengarahan dan semangat bagi penulis dalam menempuh masa perkuliahan dan Tugas Akhir selama 4 tahun penuh.
5. Bapak Dr. Ir. Febriliyan Samopa, S.Kom., M.Kom. selaku dosen pembimbing yang begitu sabar memberikan bimbingan, dukungan, semangat, dan motivasi dalam menyelesaikan Tugas Akhir ini dengan baik dan benar.
6. Bapak Nanok Adi Saputra, selaku laboran dari Laboratorium Infrastruktur dan Keamanan Teknologi Informasi (IKTI) yang senantiasa membantu penulis dalam hal administrasi penyelesaian Tugas Akhir dan selalu memastikan laboratorium nyaman untuk dipakai seluruh mahasiswa lab. IKTI.
7. Para Bapak dan Ibu dosen Jurusan Sistem Informasi ITS.
8. Sahabat-sahabat terbaik penulis Dwi Karya Maha Putra, Daniel Surya Anjas, Bryan Damanik, Gresela Sitorus, Yani Surbakti, Febe Panjaitan, Belli Silaban, Kevin Simanjuntak, dan Harapan Daeli atas semangat, dukungan, dan kebersamaannya selama perkuliahan
9. Teman-teman angkatan 2013 Sistem Informasi, BELTRANIS yang menjadi keluarga bagi penulis selama 4 tahun perkuliahan ini.
10. Seluruh Staff dan karyawan di Jurusan Sistem Informasi, yang telah bekerja sebaik mungkin dalam membantu penulis dalam menyelesaikan urusan akademik selama perkuliahan maupun tugas akhir.

Penelitian ini diharapkan dapat menjadi acuan yang baik dalam melakukan pengembangan aplikasi yang menjadi salah satu focus jurusan Sistem Informasi. Penulis menyadari masih terdapat banyak kekuurangan dalam penelitian dan penyusunan Buku Tugas Akhir ini, oleh karena itu penulis menerima dengan sangat terbuka apabila ada kritik dan saran yang membangun untuk semakin menyempurnakan penelitian tugas akhir ini. Semoga dengan selesainya tugas akhir ini, banyak pihak yang semakin terbantu

Surabaya, 18 Januari 2018

Penulis

(Halaman Sengaja Dikosongkan)

DAFTAR ISI

(Halaman Sengaja Dikosongkan)

DAFTAR TABEL

(Halaman Sengaja Dikosongkan)

DAFTAR GAMBAR

(Halaman Sengaja Dikosongkan)

BAB I
PENDAHULUAN

Pada bab pendahuluan ini akan dijelaskan mengenai latar belakang masalah perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan relevansinya terhadap jurusan sistem informasi. Uraian dari bab ini diharapkan mampu memberikan keterangan mengenai gambaran umum permasalahan dan pemecahan yang diharapkan.

1.1. Latar Belakang Masalah

Meningkatnya kebutuhan manusia memberikan dampak positif bagi pertumbuhan pasar. Para perusahaan/distributor khususnya akan mendapat banyak pesanan dari para pengecer dan jenis pedagang lain dengan jumlah permintaan produk yang besar. Jumlah permintaan yang besar dari banyak pelanggan mengharuskan perusahaan mengirimkannya menggunakan banyak armada dengan tidak mengesampingkan efisiensi. Demi mewujudkan efisiensi ini, pihak manajemen perusahaan harus mengoptimalkan jumlah armada yang dimilikinya dengan jumlah produk yang dikirim, lokasi pengiriman, dan waktu pengirimannya. Namun saat ini, perusahaan masih kesulitan dalam memonitor kinerja armadanya, apakah sedang bekerja, apakah mengantarkan barang pada jalur yang seharusnya, dan apakah barang yang sampai sesuai dengan yang dibawa dari gudang. Perusahaan juga akan sangat terbantu apabila pelanggannya bisa memberikan konfirmasi secara langsung saat produk yang dipesan sudah tiba.

Untuk memenuhi kebutuhan tersebut, perusahaan dapat menggunakan aplikasi yang memanfaatkan teknologi *smartphone* agar bisa digunakan secara *mobile*. Dengan aplikasi "*Sistem Pemantau Armada*" dari penelitian ini, manajemen perusahaan dapat memberikan tugas pengiriman untuk para sopir armada pengangkut dengan cepat dan tentunya dapat melihat lokasi sopir secara *realtime*, sopir dapat menerima tugas

dengan cara yang mudah juga dapat memberikan lokasi terkininya kepada pihak manajemen perusahaan, dan pelanggan dapat memberikan konfirmasi barang pesanan yang sudah diterima dan belum diterima kepada perusahaan. Penggunakan aplikasi ini tentu berdampak positif dalam menyelesaikan masalah perusahaan seperti yang dijelaskan diatas. Namun aplikasi ini juga memiliki dampak negatif yakni semua pengguna aplikasi ini harus menggunakan *smartphone* yang *compatible* dengan aplikasi yang tersedia, dan juga harus selalu terkoneksi dengan internet dan fitur *Global Positioning System* (*GPS*).

Sistem Pemantau Armada yang sudah mulai diaplikasikan pada beberapa perusahaan yang memiliki armada kendaraan. Bahkan sudah ada perusahaan khusus untuk membuat sistem pemantauan armada ini seperti *Nextraq*, *Capterra*, *RhinoFleettracking*, *Ruptella*, dsb. Ada juga pembuatan sistem pemantau armada dari hasil penelitian-penelitian. Namun dari kesemuanya itu, sistem yang dibuat masih berupa alat yang dipasangkan dalam kendaraan armada, lalu alat tersebut terhubung ke *server*. Perusahaan kemudian akan mendapatkan data-data terkait armada dari *server*. Sistem yang dibuat oleh perusahaan khusus tersebut bisaanya tidak hanya memantau lokasi saja, namun lebih luas lagi seperti konsumsi bahan bakar, kecepatan, saran rute paling efisien, dan sebagainya. Tidak seperti penelitian Tugas Akhir ini yang tidak memasangkan alat pada kendaraan namun lebih tepatnya memantau lokasi *smartphone* yang digunakan setiap sopir armada perusahaan karena memang fokus penelitian ini hanya pada pemantauan lokasi armada.

Berdasarkan *Oxford Dictionaries*, sebuah telepon genggam dikatakan *smartphone* apabila memiliki layar sentuh, akses internet, dan memiliki sistem operasi yang memungkinkan untuk menjalankan aplikasi yang telah diunduh. [1] Lebih luasnya *smartphone* adalah kombinasi dari *cellphone* dan *handheld computer* yang menghubungkan antara *web*, musik,

pemutar musik, kamera, *GPS*, pengenal suara, pencarian dengan suara, dan banyak lagi. Karena banyaknya fitur yang disediakan dalam 1 perangkat kecil inilah, orang-orang tertarik untuk menggunakannya. Pasar *smartphone* saat ini berkembang sangat pesat. Hal ini terlihat dari hasil survei yang diadakan statista, jumlah pengguna *smartphone* di dunia pada tahun 2014 mencapai 1,57 miliar dan diprediksi bisa mencapai 2,87 miliar pengguna di tahun 2020 mendatang. Google *Android* dan Apple iOS adalah sistem operasi (OS) *smartphone* yang paling banyak digunakan saat ini. [2] Dari total yang terjual di kuarter pertama 2015, 78.8% *smartphone* menggunakan OS *Android*, 17.9% menggunakan iOS, dan sisanya menggunakan OS lainnya seperti Nokia Siymbian, RIM Blackberry OS, dan Microsoft Windows Phone. Angka tersebut sedikit berubah pada kuarter ketiga 2016 dimana 87.8% yang terjual adalah *Android* sedangkan iOS turun menjadi 11.5%. [3] Hal ini menunjukkan saat ini *smartphone Android* dan Apple masih memiliki potensi yang sangat baik untuk dikembangkan.

Diantara 2 OS besar saat ini, penelitian ini akan membuat aplikasi yang berbasis *Android* karena selain persentase penggunaan aplikasi yang jauh lebih besar dibandingkan *iOS*, harga *smartphone* yang menggunakan *Android* lebih murah dan banyak variannya sehingga cocok untuk kalangan atas dan bawah. Contoh perbandingan harga untuk penjualan di Indonesia adalah harga versi terendah *Apple iPhone* saat ini yakni iPhone 5s 16GB adalah Rp 2.349.000 dan versi tertingginya yakni *iPhone 7 Plus* 128GB adalah Rp 13.150.000. [4] Sementara *smartphone Android* sudah bisa didapat dengan harga Rp 1.000.000. Alasan lain adalah untuk menjadi seorang developer Apple program, maka harus membayar USD 99 per tahunnya agar bisa mendistribusikan aplikasinya sementara *Android* tidak memerlukan biaya apapun. [5] Dalam pengembangan aplikasi Sistem Pemantau Armada, akan ada sedikit tantangan dalam pengerjaan mengingat banyaknya variasi spesifikasi *smartphone* yang berbasis *Android* dimana aplikasi diharuskan dapat beroperasi dalam perangkat dengan

spesifikasi rendah dan spesifikasi tinggi, dengan layar yang kecil maupun layar yang besar. Aplikasi terutama harus bisa dijalankan di *Smartphone* dengan kisaran harga dibawah Rp 2.000.000 mengingat sopir tidak menggunakan handpone yang mahal.

1.2. Perumusan Masalah

Dari uraian latar belakang diatas, maka rumusan masalah yang menjadi fokus untuk diselesaikan dalam Penelitian Tugas Akhir ini adalah:

1. Bagaimana memantau lokasi armada/kurir barang secara *realtime*?
2. Bagaimana memantau kinerja armada yang berbeda untuk mengerjakan tugas yang sama?
3. Bagaimana memberikan penugasan pada armada untuk mengantarkan pesanan pada pelanggan secara mudah dan cepat?
4. Bagaimana memantau keberhasilan dalam memenuhi pesanan pelanggan?

1.3. Batasan Masalah

Berdasarkan perumusan masalah diatas, batasan ruang lingkup pengerjaan tugas akhir ini adalah:

1. Maksimal perangkat yang bisa menggunakan aplikasi dalam satu waktu adalah 100 buah.
2. Menggunakan API 19: *Android* 4.4 (*KitKat*) dimana diperkirakan 90.1% perangkat *Android* dapat menggunakannya
3. Menggunakan *Google Play Services* sebagai *dependencies build.gradle versi 11.8.0*
4. Aplikasi tidak menggunakan algoritma *Vehicle Routing Problem* dan sejenisnya dalam hal pemilihan rute perjalanan sopir, melainkan memanfaatkan API *Google maps* untuk memberikan saran rute perjalanan

1.4. Tujuan Penelitian

Berdasarkan perumusan masalah yang dijelaskan sebelumnya, tujuan yang ingin dicapai dari tugas akhir ini adalah:

1. Pembuatan aplikasi *Sistem Pemantau Armada* untuk membantu manajemen perusahaan dalam memberikan tugas pada tiap armada, juga memantau lokasi dan kinerja armadanya
2. Pembuatan sistem *database* aplikasi yang terintegrasi untuk pelanggan, manajemen perusahaan, dan armada/kurir barang.

1.5. Manfaat Penelitian

Tugas akhir ini diharapkan dapat memberikan berbagai manfaat bagi berbagai pihak, yakni:

1. Bagi mahasiswa (akademisi), dapat digunakan untuk menambah pengetahuan terkait arsitektur *database*, integrasi antar perangkat yang menggunakan aplikasi, dan mengimplementasikannya dalam sebuah aplikasi berbasis *android*.
2. Bagi Jurusan Sistem Informasi ITS, menambah fortofolio terkait penelitian/ pengembangan aplikasi berbasis *android* dengan *database* yang berbasis *Firebase*.
3. Membantu dalam mendefinisikan spesifikasi kebutuhan dengan fitur-fitur yang tepat dari aplikasi "*Sistem Pemantau Armada*" yang dirancang sesuai dengan kemampuan pemakai
4. Membantu dalam mendefinisikan desain antar muka yang tepat sesuai dengan kemampuan pemakai, sehingga tanpa perlu latihan atau petunjuk khusus pengguna dapat mengoperasikan aplikasi

1.6. Relevansi

Relevansi dari penelitian ini berkaitan dengan mata kuliah Algoritma dan Pemrograman, Permrograman Berorientasi Objek, Analisa Desain Perangkat Lunak, Kontruksi

Pengembangan Perangkat Lunak, Interaksi Manusia dan Komputer, Pengantar Basis Data, Desain Basis Data, Pemrograman Berbasis Web, dan Pemrograman Perangkat Bergerak. Penelitian ini masuk ke dalam lingkup Laboratorium Infrastruktur dan Keamanan Teknologi Informasi dimana topik penelitian yang terkait pada laboratorium ini adalah transportasi dan logistik, dan desain arsitektur software.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dijelaskan mengenai penelitian sebelumnya dan dasar teori yang akan digunakan sebagai acuan dalam pengerjaan tugas akhir ini.

2.1 Studi Sebelumnya

Beberapa penelitian sebelumnya yang dapat dijadikan acuan dalam pengerjaan tugas akhir ini adalah:

Tabel 2.1 Studi sebelumnya (I)

Judul Tugas Akhir	Rancang Bangun Aplikasi *Backend* dan *Web Services* Penyampaian Laporan Masyarakat Berbasis *Crowdsourcing* melalui Jejaring Sosial Twitter
Penulis, Tahun	Degananda Ferdian Priyambada, 2016
Deskripsi umum penelitian	Pertumbuhan penduduk di Kota Surabaya semakin meningkat tiap tahunnya. Hal ini terjadi akibat banyaknya perpindahan penduduk demi mencari lapangan pekerjaan. Pertumbuhan ini menimbulkan masalah baru seperti kriminalitas, kemacetan, maupun rusaknya fasilitas-fasilitas umum. Masalah ini memberikan ide sebuah tugas akhir untuk membangun sebuah sistem yang dapat menjembatani antara masyarakat dan pemerintah Kota Surabaya sebagai pemangku kekuasaan agar dapat saling menyampaikan laporannya. Melalui aplikasi ini masyarakat dapat menyampaikan laporannya melalui situs surabaya113.com yang isinya berupa permasalahan dan bukti foto. Meski ada fasilitas ini, masyarakat masih jarang menggunakannya dan cenderung menyebar

	informasi ke media Twitter. Pada sisi *back*end tugas akhir ini juga akan menyertakan peta persebaan laporan masyarakat Surabaya yang diambil dari jejaring sosial Twitter.
Keterkaitan penelitian	Tugas akhir ini membuat sebuah aplikasi dimana desain aplikasinya menggunakan *Google Design Sprint*. Tahapan Design Sprint dilakukan mulai dari understand (Masalah dan solusi, komponen, atribut komponen), define (fitur), diverge (*tidak dilakukan*), decide, *prototype* (*User* Stories, Epic dan skenario, storyboard), validate (validasi)

Tabel 2.2 Studi sebelumnya (II)

Judul Paper	*Mapping Vessel Path of Marine Traffic Density of Port Klang, Malaysia using Automatic Identification System (AIS) Data*
Penulis, Tahun	Masnawi Mustaffa, Noor Hasnita Ahmat, Shaharuddin Ahmad, 2013
Deskripsi umum penelitian	AIS Technology adalah sebuah sistem yang memungkinkan sebuah kapal untuk mendapatkan informasi dari kapal yang d*item*uinya mengenai posisi, jalannya, kecepatan, dll secara otomatis dengan very high frequency (VHF) dari transimisi radio. Paper ini membahas analisa dari data AIS yang dikumpulkan untuk memeriksa pergereakan kapal yang transit di Pelabuhan Klang dan Selat Malaka dengan memetakan kapal berdasarkan posisinya (*latitude*, *longitude*). Pemetaan diinvestigasi dan dievaluasi untuk memvisualisasikan alur dari lalu lintas kapal dengan AIS

	technology. Hasilnya adalah terlihat lalu lintas yang padat di sepanjang Selat Malaka dan di Pelabuhan Klang. Selat ini adalah jalur kapal utama antara Samudra Hindia dan Samudra Pasifik, yang menghubungkan ekonomi Asia yakni India, Cina, Jepang, dan Korea Selatan
Keterkaitan penelitian	Paper ini berfokus pada bagaimana Automatic Identification System (AIS) data dapat digunakan untuk mempelajari lalu lintas kapal di jalur yang padat. AIS menyediakan data yang akurat secara *realtime*. Dengan AIS posisi setiap kapal di daerah Pelabuhan Klang dan Selat Malaka dapat diketahui.

Tabel 2.3 Studi sebelumnya (III)

Judul Paper	*Real-Time Tracking Manajement System Using GPS, GPRS, and Google Earth*
Penulis	Noppadol Chadil, Apirak Russameesawang, Phongsak Keeratiwintakorn
Deskripsi umum penelitian	Karena tingginya harga energi fosil, harus dilakukan metode untuk mengurangi penggunaan energi dalam logistik dan armada. *GPS tracking* adalah salah satu pendekatan untuk mendapatkan informasi lokasi kendaraan secara *realtime*. Dalam paper ini peneliti membuat *GPS tracking* menggunakan *commodity hardware*, *software open source*, dan UI yang mudah melalui *web server* dengan *Google map* atau Google earth. Sistem menggunakan modul GPS untuk akuisisi lokasi dan transmisi pesan, MMC untuk menyimpan informasi lokasi sementara, dan 8-bit AVR

	microcontroller. System diuji coba dalam sebuah perjalanan dari Bangkok ke Chonburi. Hasilnya terlihat bahwa alat stabil dan protokol transfer pesannya juga kuat sehingga hampir semua lokasi dapat diketahui dan ditransmisikan secara *realtime*.
Keterkaitan penelitian	Penelitian dalam paper ini bertujuan untuk melakukan *tracking* kendaraan secara *realtime* dengan menggunakan *Google maps* melalui sebuah alat (8-bit AVR RISC Microcontroller) dengan UART, SPI, dan 12C agar bisa terhubung dengan GPRS dan *GPS module*. Kemudian alat dipasang pada kendaraan untuk mendapatkan informasi lokasi. Kemudian Informasi tersebut dikirimkan melalui GPRS ke *Server*, dan *server* menyimpannya dalam *database*.

Tabel 2.4 Studi sebelumnya (IV)

Judul Paper	*Fuel Monitoring and Vehicle Tracking Using GPS, GSM, and MSP430F149*
Penulis, Tahun	Sachin S. Aher, Kokate R. D. , 2012
Deskripsi umum penelitian	Sampai saat ini, pengelolaan bahan bakar yang diisi dan dikonsumsi tiap kendaraan tidak diperhatikan sehingga berpengaruh pada pemborosan finansial. Untuk mengatasi hal tersebut, penelitian dalam paper ini membuat sebuah microcontroller untuk memantau bahan bakar dan sistem pelacakan kendaraan. Menggunakan reed switch, pengguna dapat membandingkan antara bahan bakar yang sebenarnya dikonsumsi dengan bahan bakar yang ada di tangki kendaraan. Kemudian dengan

	teknologi GPS, *user* juga dapat memantau lokasi kendaraan. Dengan alat ini, perusahaan dapat memantau bahan bakar, lokasi kendaraan, bahkan dimana lokasi saat bahan bakar paling banyak digunakan. Dengan reed switch, perusahaan juga dapat mendeteksi apabila ada pencurian bahan bakar dan dimala lokasi pencuriannya dilakukan.
Keterkaitan penelitian	Penelitian dalam paper ini bertujuan utama untuk memantau penggunaan bahan bakar pada kendaraan. Untuk lebih jauhnya perusahaan juga ingin mengetahui dimana lokasi yang paling banyak menghabiskan bahan bakar dan mengetahui lokasi dan waktu apabila ada pencurian bahan bakar kendaraan.

2.2 Dasar Teori

Bagian ini akan menjelaskan teori dan bahan penelitian lain yang menjadi ilmu dasar dalam mengerjakan tugas akhir ini.

2.2.1 *Sistem Pemantau Armada*

Suatu perusahaan besar yang bergerak dalam produksi maupun penyaluran barang membutuhkan banyak armada pengiriman produk untuk tiap pelanggannya. Banyaknya armada ini tentu akan semakin menyulitkan perusahaan dalam mengontrol kinerjanya, baik dimana lokasi armada secara *realtime*, kinerja armada (ketepatwaktuan dan displin), maupun pemberian penugasan armada dengan mudah. Oleh karena itu dibutuhkan suatu sistem yang bisa membantu perusahaan menyelesaikan masalah tersebut. *Sistem Pemantau Armada* adalah suatu produk untuk memantau baik kinerja, lokasi, maupun informasi lainnya sesuai dengan kebutuhan pelanggan. Bagi perusahaan yang menggunakannya, sistem ini diharapkan dapat memaksimalkan produktivitas, mengurangi biaya,

meningkatkan kepuasan pelanggan, dan meningkatkan pendapatan layanan perusahaan.

Sistem ini dapat dibangun dalam berbagai jenis *platform* seperti *smartphone*, *desktop*, maupun perangkat keras lainnya yang dikostumisasi sesuai dengan spesifikasi sistem yang dibangun. Dengan kebutuhan yang diharapkan pelanggan tersebut, sistem tentu harus menggunakan teknologi Global Positioning System (GPS) untuk melacak lokasi armada, dan sistem *database* yang terintegrasi sehingga tiap armada dapat menyimpan riwayat perjalanannya.

Saat ini *Sistem Pemantau Armada* sudah banyak digunakan di perusahaan. Berikut adalah beberapa contoh produk yang berhasil dikembangkan:

A. Trackimo

Trackimo adalah sebuah aplikasi yang memanfaatkan 3*G GPS Fleet tracking system*. Dalam penggunaanya Trackimo merupakan sebuah alat kecil yang dibawa bersama armada, dan pihak manajemen dapat mengaksesnya melalui situs web. Produk ini memberikan beberapa kelebihan bagi perusahaan seperti mengurangi konsumsi bahan bakar dan waktu tempuh dengan memberikan saran rute terdekat dan terhindar dari macet. Lalu dengan fitur remote employee, *activity* seperti *idling* mesin berlebihan, penggunakan yang tidak terotorisasi, *overtime*, kecepatan tinggi, dan sebagainya yang dapat membahayakan dan merugikan perusahaan dapat dihindari. Kemudian fitur *reduce downtime* yang memberikan informasi mengenai armada perusahaan mana yang dalam keadaan *idling*, sehingga membantu perusahaan untuk membuat strategi manajemen armada yang lebih baik lagi. [6]

B. Teletrac Navman

Teletrac Navman adalah sebuah aplikasi yang memanfaatkan GPS untuk mengetahui status armada

setiap waktu, seperti dimana keberadaannya, apakah dalam perjalanan, apakah berhenti, maupun dalam jalanan macet. Dengan informasi ini, pihak manajemen perusahaan dapat mengetahui dengan tepat waktu produk tiba, maupun merubah merubah rute dalam bagian perjalanan, dan berkomunikasi dengan sopir. Semua informasi tersebut dapat diakses oleh perusahaan melalui *website*. Produk ini juga menawarkan efisiensi bahan bakar, yang tidak hanya mengefisiensikan jarak tempuh, namun menghindari lalu lintas padat. Kemudian fitur *driver monitoring* yang memungkinkan untuk merekam *activity* sopir seperti kecepatan, pengereman yang membahayakan, percepatan kendaraan yang tinggi, dan kendaraan yang tersudut. Fitur ini dapat memberikan penilaian kinerja sopir sehingga dapat mengurangi risiko kecelakaan dan membantu sopir dalam mempertahankan karir yang baik dalam pekerjaannya. [7]

C. *Rhino Fleet Tracking*

Rhino Fleet Tracking adalah sebuah sistem yang memberikan layanan pelacakan armada dalam jumlah banyak. Produk ini memberikan penawaran seperti pelacak kendaraan, pelacak kereta, pelacak perlengkapan, dan pelacak daya baterai. Pelacakan armada dalam jumlah berpapun akan tetap menyulitkan, tapi proses pelacakan dipermudah dengan *Rhino Fleet Tracking's Fleet telematic solution*, sehingga memberikan akses aman pada informasi kendaraan. Sistem ini juga melakukan kerjasama dengan *Amazon Alexa*, yang memungkinkan *user* untuk mendapatkan data *Rhino Fleet Tracking* hanya dengan perintah suara.

Proses pelacakan kendaraan dimulai dari proses transmisi data dari alat yang ada di kendaraan pada *GPS receiver*, kemudian diterjemahkan menjadi *monitoring data* dan ditampilkan secara visual dengan bantuan *Google map*. [8]

Dalam pembuatan Sistem Pemantau Armada yang berbasis *Android* ini nantinya akan dibangun 2 aplikasi yang berbasis *Android*. 1 aplikasi dirancang untuk digunakan pihak manajemen perusahaan. Aplikasi ini memiliki fitur untuk mengecek armada dalam kondisi *idling*, memberikan penugasan pada sopir berdasarkan pesanan yang masuk, posisi dan riwayat posisi kendaraan yang sedang membawa barang beserta sopir yang membawa, barang yang dibawa, dan kemana tujuannya. Fitur lainnya adalah bisa melihat riwayat performa sopir yang melakukan tugas yang sama. Sementara 1 aplikasi lagi dirancang untuk digunakan oleh sopir yang mengendarai armada. Aplikasi ini memiliki fitur untuk menerima tugas pengiriman barang dari pihak manajemen perusahaan, mengirimkan riwayat lokasi kepada manajemen perusahaan, dan memberikan konfirmasi barang yang sudah sampai di kepada pemesan. Selain 2 aplikasi berbasis *Android*, *Sistem Pemantau Armada* ini juga memiliki fitur *website* untuk menampilkan Peta persebaran armada dan riwayat lokasinya. Sehingga riwayat tersebut dapat diakses melalui *desktop* dan memungkinkan untuk ditampilkan di layar yang berukuran besar.

Terdapat bebrapa aktor yang berperan dalam sistem pemantaan kendaraan ini . Berikut adalah daftar aktor dan fungsinya dalam proses penanganan laporan:

A. Manajemen perusahaan
Manajemen perusahaan adalah aktor yang memiliki akses untuk mendapatkan fungsi utama dari aplikasi *Fleet Monitoring System* yakni mengelola pesanan pelanggan, memberikan penugasan pada setiap armada yang terdaftar dalam perusahaan, melihat ketersediaan setiap armada, dan memantau lokasi setiap armada yang sedang bertugas.

B. Sopir armada
Sopir armada adalah aktor yang memiliki tugas utama untuk mengirimkan lokasi mereka pada *server* sehingga

manajemen perusahaan dapat mengecek lokasi mereka. Sopir disini berperan sebagai penerima tugas pengiriman barang dan mengirimkan barang sampai ke pelanggan.

C. Pelanggan

Pelanggan adalah aktor yang memiliki tugas untuk memberikan pesanan dan mengkonfirmasi barang apa saja dari pesanan mereka yang sudah sampai. Konfirmasi ini akan dikirimkan ke manajemen perusahaan sebagai referensi untuk perusahaan menyelesaikan keseluruhan pesanan pelanggan.

Berikut adalah proses bisnis sebagai acuan pembuatan sistem ini:

A. Manajemen Perusahaan

Pada awalnya perusahaan harus memiliki identitas seperti deskripsi perusahaan, barang yang dijual, daftar armada dengan deskripsinya, dan daftar sopir dengan deskripsinya. Dalam praktiknya sistem ini tidak mengatur secara otomatis penjadwalan dan pemilihan sopir dari setiap pesanan yang ada, namun dilakukan secara manual oleh manajemen perusahaan. Jadi proses-proses yang dilakukan adalah:

Tabel 2.5 Proses bisnis manajemen perusahaan (I)

Proses	Input	Output
Membuat penugasan Baru	Data pesanan (ID Produk, Nama produk, jumlah produk, nama pelanggan, alamat pelanggan)	Data penugasan (ID Produk, Nama produk, jumlah produk, nama pelanggan, alamat pelanggan, Token, ID penugasan)

Mengecek Armada yang Tersedia	Daftar Sopir (ID Armada, Nama armad)	ID Armada terpilih
Memberikan Penugasan pada Sopir Armada	ID Armada, Data penugasan	Mapping penugasan (ID Armada, ID penugasan, ID Produk, Nama produk, jumlah produk, nama pelanggan, alamat pelanggan)

Tabel 2.6 Proses bisnis perusahaan (II)

Proses	Input	Output
Konfirmasi barang sudah diterima	ID penugasan, ID Armada, ID Produk diterima, Jumlah Pengiriman	-
Memberi status tugas selesai pada sopir	ID Armada, Data penerimaan	Status Penugasan "Selesai"
Memberi notifikasi sopir untuk kembali	ID Armada, Data penugasan, Data penerimaan	Status penugasan "ambil barang (ID Penugasan)"
Memberi penugasan tambahan	ID Armada, Data Penugasan (tambahan)	Mapping penugasan (ID Armada, ID Produk, Nama produk, jumlah produk, nama pelanggan, alamat

		pelanggan, ID penugasan)

B. Sopir Armada

Sopir armada tidak mendaftarkan diri dari perangkat yang dimilikinya melainkan dari perangkat yang dimiliki perusahaan. Pada perangkatnya sopir hanya bisa log in menggunakan *username* dan *password* yang diberikan perusahaan.

Tabel 2.7 Proses bisnis Sopir armada (I)

Proses	**Input**	**Output**
Mengkonfirmasi pengerjaan tugas	Daftar Mapping Penugasan	ID Pesanan, Status Penugasan "Dilakukan"
Mengirimkan lokasi Armada	*Latitude*, *Longitude* dari GPS	*Latitude*, *Longitude* terkirim ke *server* secara periodik
Mengirimkan barang ke Pelanggan	-	-
Memberikan form Penerimaan barang	-	-

Tabel 2.8 Proses bisnis sopir armada (II)

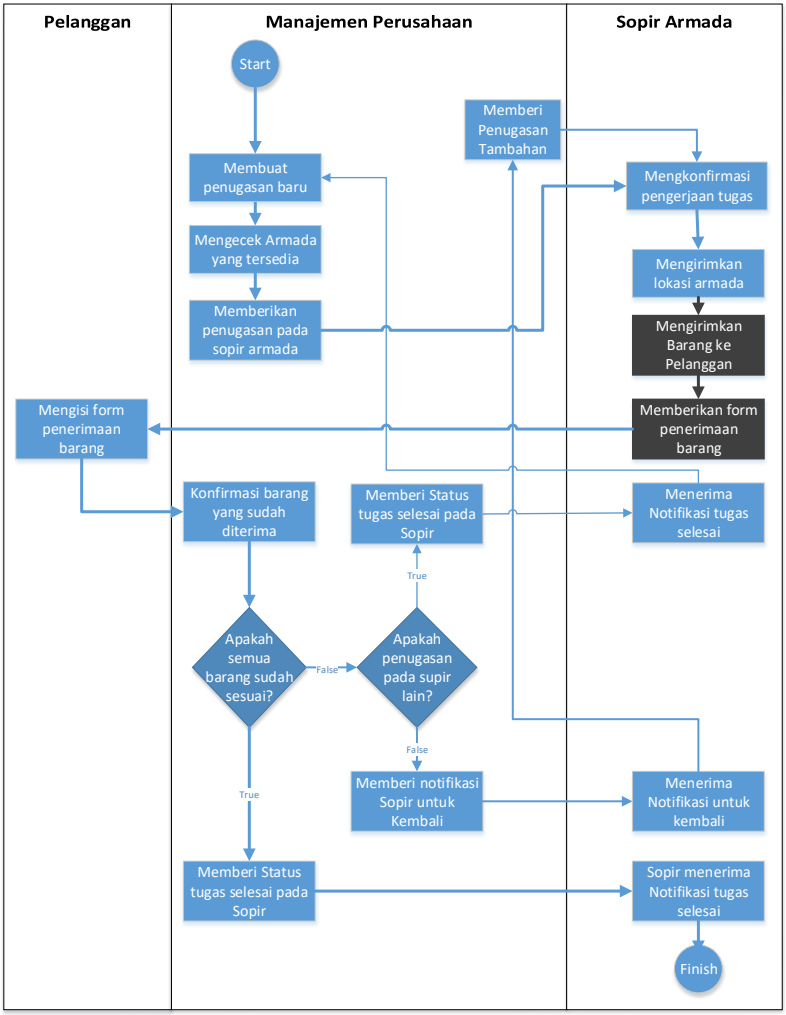
Proses	Input	Output
Menerima notifikasi tugas selesai	Status Penugasan "selesai"	-
Menerima notifikasi untuk kembali	Status Penugasan "Ambil barang (ID Penugasan)"	-

C. Pelanggan

Pelanggan tidak memiliki perangkat khusus untuk dapat menggunakan sistem ini karena pelanggan tidak melakukan pemesanan menggunakan aplikasi Sistem Pemantau Armada. Untuk dapat mengkonfirmasi pesanan yang diterima, pelanggan dapat mengkonfirmasinya melalui perangkat yang dimiliki sopir dan mengisikan sebuah token yang diberikan perusahaan untuk memastikan validitas data yang diberikan.

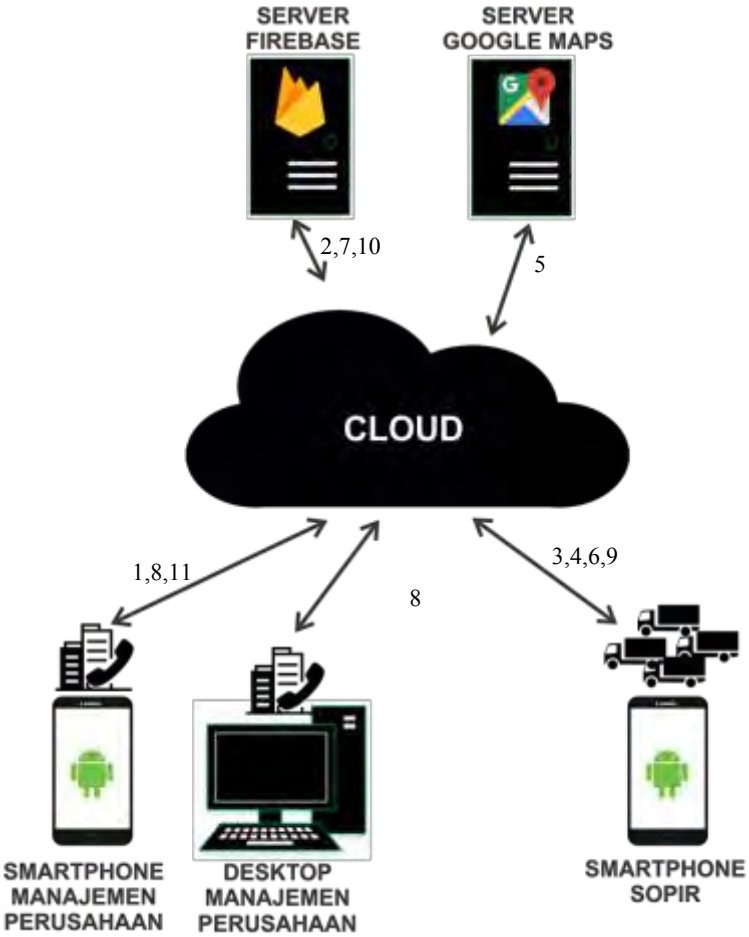
Tabel 2.9 Proses bisnis pelanggan

Proses		Input	Output
Mengisi *Form* Penerimaan barang		ID Produk diterima, Nama produk diterima, jumlah produk diterima, Token, ID Armada, ID Penugasan)	Data penerimaan (ID Produk diterima, jumlah produk, Token, Armada, ID Penugasan)



Gambar 2.1 Konsep proses bisnis aplikasi

Dari proses bisnis tersebut, arsitektur perangkat lunak dari Sistem Pemantau Armada dikembangkankan dengan beberapa teknologi seperti yang dapat dilihat pada Gambar 2.2 sebagai berikut:



Gambar 2.2 Konsep arsitektur system

Tabel 2.10 Penjabaran konsep arsitektur sistem

No	Proses	Data
1	Mengisi data penugasan Sopir Armada ke Cloud	ID Armada, Data penugasan (ID Produk, Nama produk, jumlah produk, nama pelanggan, alamat pelanggan, Token, ID Pesanan)
2	Cloud menyimpannya di *server Firebase*	Mapping penugasan (ID Armada, ID penugasan, ID Produk, Nama produk, jumlah produk, nama pelanggan, alamat pelanggan)
3	*Smartphone* Sopir menerima data penugasan dari Cloud yang dikirimkan *Firebase*. Sopir menjalankan pekerjaan dan mengaktifkan fitur pelacakan lokasi	Mapping Penugasan
4	Smarphone Sopir mengaktifkan GPS dan meminta kordinat lokasi ke Cloud	API Key
5	*Smartphone* Sopir menerima kordinat dari cloud yang dikirimkan *Server Google maps*	API Key, *Latitude*, *Longitude*
6	*Smartphone* Sopir mengirimkan kordinat lokasi ke Cloud untuk disampaikan ke *Firebase*	ID Perusahaan, ID Armada, *Latitude*, *Longitude*
7	Cloud mengirimkan data dari *Smartphone* Sopir ke *Firebase*	ID Perusahaan, ID Armada, *Latitude*, *Longitude*

8	Perangkat manajemen perusahaan yakni *smartphone* dan *desktop* menerima kordinat lokasi Sopir dari Cloud yang dikirimkan dari *Firebase*	ID Perusahaan, ID Armada, *Latitude*, *Longitude*
9	Saat penugasan telah selesai, *Smartphone* sopir mengirimkan konfirmasi ke Cloud	ID Perusahaan, ID Armada, Data Penugasan, Status Penugasan
10	Cloud mengirimkan Data penugasan ke *Server Firebase*	ID Perusahaan, ID Armada, Data Penugasan, Status Penugasan
11	*Smartphone* menerima konfirmasi penugasan dari Cloud yang dikirimkan *Firebase*	ID Perusahaan, ID Armada, Data Penugasan, Status Penugasan

2.2.2 *Android*

Sistem operasi yang akan digunakan untuk pengembangan aplikasi *Fleet Monitor*ing ini adalah *Android* yang merupakan sistem operasi open source yang berbasis Linux, dengan platform sebagai berikut: [9]

Gambar 2.3 Android – Platform

1. **Linux Kernel**
   Android mengadopsi Linux Kernel untuk menjadi lapisan paling dasar *Android*. Linux Kernel ini berfungsi sebagai basis untuk manajemen proses, manajemen memori dan manajemen perangkat. Kernel juga menyediakan berbagai driver untuk semua hardware yang ada di smarphone agar aplikasi dapat dengan mudah mengakses dan menggunakan hardware. Dengan kata lain, Kernel berfungsi sebagai jembatan antara software dan hardware.
2. **Hardware Abstraction Layer (HAL)**
   HAL menyediakan antarmuka standar yang mengekspos kemampuan hardware ke level Java API Framework yang lebih tinggi. HAL terdiri dari beberapa modul *library* yang masing -masingnya berguna untuk mengimplementasikan interface untuk komponen hardware yang lebih spesifik seperti module kamera atau Bluetooth. Saat kerangka API mengakses hardware, *Android* system akan memanggil modul perpustakaan untuk komponen hardware yang dibutuhkan.
3. ***Android* Runtime**
   Untuk perangkat dengan versi *Android* 5.0 (API Level 21), setiap aplikasi berjalan atas proses sendiri dengan instance sendiri dari *Android* Runtime (ART). ART dibuat untuk menjalankan beberapa virtual machine sekaligus pada low-memory devices dengan mengeksekusi DEX (Dalvic Executable) files, sebuah bytecode format yang didesain untuk mengoptimasi penggunaan memory.
4. **Native C/C++ libraries**
   Banyak komponen utama *android* system seperti HAL dan ART, dibangun dengan native code yang membutuhkan native *library* berbasis C atau C++. Platform *Android* menyediakan java frameworks APIs untuk menampilkan fungsionalitas dari *library* ini ke aplikasi. Misalkan, kita dapat mengakses OpenGL ES

pada *Android* framework's Java OpenGL API untuk menambahkan dukungan menggambar dan memanipulasi grafik 2D dan 3D dalam aplikasi. Jika ingin membangun aplikasi yang membutuhkan kode C/C++, kita dapat menggunakan *Android* NDK untuk mengakses native platform libraries langsung dari kode kita.

5. **Java API Framework**
   Semua fitur dari sistem operasi *Android* tersedia untuk kita melalui API yang dituliskan dalam Bahasa pemrograman Java. API ini membentuk *building block* yang dibutuhkan untuk membangun aplikasi *Android* dengan menyederhanakan penggunaan kembali inti, komponen dan servis dari modular sistem, yang meliputi:
   - View System yang dapat digunakan untuk membangun *User interface*
   - Resource Manager yang memungkinkan akses ke resource non-code seperti string, grafik dan layout files
   - Notification Manager yang memungkinkan semua aplikasi menampilkan peringatan di status bar
   - Activity Manager yang mengatur lifecycle dari aplikasi
   - Content Providers yang memungkinkan aplikasi untuk mengakses data dari aplikasi lain seperti kontak atau mengirimkan data mereka
6. **System Apps**
   Pada dasarnya, *Android* sudah menyediakan aplikasi inti seperti *email*, SMS, kalender, internet browser, kontak, dll. Namun beberapa diantara aplikasi yang disediakan ini tidak terlalu digunakan pengguna. Oleh Karena itu aplikasi pihak ketiga menjadi pilihan para *user*.

Dari keenam komponen diatas, komponen yang paling dikenal adalah System Apps Karena itulah aplikasi-aplikasi yang terlihat dan digunakan oleh para *user*. Aplikasi juga dapat dikembangkan secara bebas dan gratis oleh para pengembang (*developer*) menggunakan *Android Software Development Kit*, baik *Android* Studio yang disediakan langsung dari Google maupun menggunakan IDE lainnya. Dengan tersedianya development platform gratis, pengembang Andoid dapat membangun aplikasi yang memanfaatkan penuh fitur *smartphone* seperti akses informasi lokasi, menjalankan *background services*, membuat alarm, membuat notifikasi di status bar, dll.

2.2.3 *Firebase*

Firebase adalah sebuah alat dan infrastruktur untuk *mobile* dan *web application* yang didesain untuk mendukung pengembangan aplikasi. *Firebase* memiliki 4 fitur penawaran utama untuk pelanggan yakni *Analytics*, *Develop*, *Grow*, dan *Earn*. [10] Untuk penyelesaian Tugas Akhir ini, fitur *firebase* yang digunakan adalah *Realtime Database* yang merupakan bagian dari *Develop*. *Realtime database* menyajikan fitur *database* cloud tanpa *SQL*. Data disimpan dalam bentuk *JSON*, disinkronisasikan melalui perangkat yang terkoneksi, dan tetap dapat digunakan meski aplikasi tidak terhubung dengan internet.

2.2.3.1 Struktur Data

Data dalam *Firebase realtime database* disimpan dalam bentuk *JSON* object. Semua data disimpan dalam cloud dalam bentuk *JSON tree*, sehingga tidak seperti *SQL* yang menyimpan data dalam bentuk tabel. Saat menambahkan data dalam *JSON* tree, maka data menjadi sebuah node dalam struktur *JSON* dan memiliki key tersendiri yang bisa kita buat dengan *user* ID atau semantic names. Bentuk *JSON* tree untuk menyimpan data dapat dilihat dari skema di bawah ini:

```
{
  "users": {
    "alovelace": {
      "name": "Ada Lovelace",
      "contacts": { "ghopper": true },
    },
    "ghopper": { ... },
    "eclarke": { ... }
  }
}
```

Gambar 2.4 Firebase - Contoh JSON Tree

Skema diatas menunjukkan bahwa tree "*user*s" memiliki 3 data, yakni "alovelace", "ghopper", dan "eclarke". Isi alovelace memiliki 2 jenis data di dalamnya yang diberikan id "name" dan "contacts". [11]

2.2.3.2 Membaca dan Menulis Data pada *Realtime Database*

Secara default, akses untuk membaca dan menulis pada *database* hanya terbatas untuk *user* yang terotentikasi. Akses tersebut dapat diatur agar akses tanpa ada izin otentikasi yakni mengubah autran menjadi public access. [12] *Firebase* databae menggunakan asynchronous *list*ener untuk memperoleh data. *List*ener digunakan pada initial state dan setiap ada perubahan data. Agar bisa membaca dan menulis data, harus ada *Database*Reference yang didefinisikan sebagai berikut:

```
private DatabaseReference mDatabase;
mDatabase =
FirebaseDatabase.getInstance().getReference();
```

Gambar 2.5 Firebase – DatabaseReference

Beberapa tipe data yang bisa digunakan pada *JSON* adalah: String, Long, Double, Boolean, Map <String, Object>, dan *List*

<Object>. Misalkan kita akan menambahkan *user*, kita dapat menggunakan operasi dasar setValue() seperti berikut:

```
private void writeNewUser(String userId, String
name, String email) {
    User user = new User(name, email);
    mDatabase.child("users").child(userId).setVa
lue(user);
}
```

Gambar 2.6 Firebase - contoh setValue() 1

Menggunakan setValue() dengan cara diatas akan menimpa data pada lokasi tertentu beserta child nodes di dalamnya. Sehingga apabila ingin melakukan update child tanpa menimpa data sebelumnya dapat menggunakan cara sebagai berikut:

```
mDatabase.child("users").child(userId).child("use
rname").setValue(name);
```

Gambar 2.7 Firebase - contoh setValue() 2

Untuk membaca data dan melihat perubahan data, gunakan method addValueEvent*List*ener() atau add*List*enerForSingleValueEvent() sehingga akan menghasilkan ValueEvent*List*ener pada *Database*Reference. ValueEvent*List*ener digunakan untuk membaca dan mendeteksi setiap perubahan pada seluruh konten data. Pada *list*ener ini, kita dapat menggunakan method onDataChange() untuk membaca konten. Method ini digunakan pertama saat *List*ener dilampirkan dan digunakan lagi setiap waktu apabila data dan anaknya mengalami perubahan.

Pada kasus tertentu, kita ingin melakukan *callback* data satu kali saja lalu kemudian dihapus, contohnya saat kita ingin menampilkan profil *user* saat membuat post baru. Dalam kasus ini data hanya perlu menggunakan data sekali dan tidak akan ada perubahan data dalam waktu dekat atau menggunakan

*list*ener yang selalu aktif. Untuk itu kita menggunakan method add*List*enerForSingleValueEvent() untuk memudahkan skenario yang dipicu sekali lalu tidak dipicu lagi.

Untuk menuliskan anak dari suatu node secara simultan tanpa menghapus anak lainnya, gunakan metod updateChildren(). Saat method ini dipanggil, kita dapat mengupdate nilai anak dengan menspesifikasi sebuah path untuk kuncinya. Kemudian untuk menghapus data dapat menggunakan method removeValue() dengan mereferensi lokasi dari data.

Fungsi *firebase* tetap dapat berjalan tanpa gangguan meskipun tanpa koneksi internet, karena setiap perangkat yang terhubung *firebase* mengelola datanya pada internal terlebih dahullu, kemudian mensikronisasikan nya dengan *database server*.

2.2.3.3 Mengoperasikan *List* Data

Saat bekerja dengan *list*, aplikasi harus bisa mendengarkan semua child event meskipun nilainya digunakan untuk single object. Child event dipicu dalam merespon operasi spesifik yang dilakukan pada chidren node seperti penggunaan method push() atau method updateChildren(). Dalam mendengar child event pada *Database*Reference, gunakan method-method **ChildEvent*List*ener** sebagai berikut: [13]

Tabel 2.11 Firebase - ChildEventListener()

Method	Penggunaan
onChildAdded()	Mengambil *list item* atau mendengarkan penambahan daftar *item*. Event ini dipicu sekali untuk setiap child yang ada kemudian setiap child baru ditambahkan ke path yang spesifik. DataSnapshot diteruskan ke *list*ener yang mengadung data child baru
onChildChanged()	Mendengarkan perubahan pada *item* dalam daftar. Event digunakan setiap

	waktu apabila child node dimodifikasi, termasuk apabila modifikasi dilakukan pada node didalamnya. DataSnapshot diteruskan ke *list*ener yang mengandung data terupdate untuk child
onChildRemoved()	Mendengarkan *item* yang dihapuskan dari *list*. DataSnaphot diteruskan ke event *List*ener yang mengandung data terupdate untuk child
onChildMoved()	Mendengarkan perubahan urutan *item* dalam daftar. Event ini dipicu apabila dilakukan update yang merubah urutan child. Event ini digunakan apabalika data diubah menggunakan orderByChild atau orderByValue

Kemudian untuk mendapatkan data secara terurut, gunakan method berikut: [13]

Tabel 2.12 Firebase - orderBy

Method	Penggunaan
orderByChild()	Mengurutkan data yang diambil berdasarkan nilai dari child key spesifik
orderByKey()	Mengurutkan data yang diambil berdasarkan child key
orderByValue()	Mengurutkan data yang diambil berdasarkan child values

Method untuk menyaring data dapat dikombinasikan dengan method order-by saat membuat query. Untuk menyaring data gunakan method: [13]

Tabel 2.13 Firebase - Filtering

Method	Penggunaan
limitToFirst()	Membatasi jumlah *item* yang akan diambil dimulai dari awal *list* data hasil
limitToLast()	Membatasi jumlah *item* yang akan diambil dimulai dari akhir *list* data hasil
startAt()	Mengambil *item* lebih banyak atau sama dengan specified key atau nilai tergantung dari order-by yang dipilih
endAt()	Mengambil *item* lebih sedikit atau sama dengan specified key atau nilai tergantung dari order-by yang dipilih
equalTo()	Mengambil *item* sesuai dengan sepcified key ata nilai tergantung dari order-by yang dipilih

2.2.4 *Google maps*

Google map API dapat digunakan pada *Android* dengan API level minimal 9, sehingga saat melakukan pengaturan proyek baru, pastikan level API *Android* sudah sesuai dengan syarat GMaps API. [14] Dengan API ini, kita dapat menambahkan peta ke aplikasi *Android* berdasarkan data yang ada di *Google maps*. API akan secara otomatis menangani akses ke *server Google maps*, mengunduh data, menampilkan peta, dan merespon gerakan peta.

Untuk menambahkan API ini, diawali dengan menambahkan Google Play Service Package yang sesuai pada *Android* Studio didalam file build.gradle, yakni:

Tabel 2.14 Google maps - API

API	Deskrpisi pada build.gradle
Google maps	com.google.*android*.gms:play-services-maps:10.0.1

Google Mobile Ads	com.google.*android*.gms:play-services-ads:10.0.1

Setelah membuat proyek baru, tambahkan satu aktivity yang menggunakan map yakni *Google maps* Activity. Setelah ditambahkan, *Android* Studio menampilkan file google_maps_api.xml yang didalamnya terdapat *Google maps* API Key. Setelah itu masuk ke link yang sudah disediakan di dalam file xml tersebut, ikuti instruksi untuk membuat proyek baru, lalu buat *Android*-restricted API Key untuk proyek ini. Salin key yang didapat lalu tambahkan di lokasi yang disediakan pada file xml. Setelah melewati langkah ini, Proyek *Android* akan bisa menggunakan API Google.

2.2.5 Development Methodology - *Google Design Sprint*

Dalam menyelesaikan suatu proyek pembuatan aplikasi, ada 2 metodologi dasar dalam pengembangan aplikasi yakni: [15]

a. *Waterfall*, yakni metodologi proses desain secara berurutan (*sequence*). Sehingga dari salah satu dari 8 tahapan metodologi ini (*conception*, *initiation*, *analysis*, *design*, *construction*, *testing*, *implementation*, dan *maintenance*), tahapan tidak bisa kembali ke tahapan sebelumnya.
b. *Agile*, yakni metotolgi yang dimulai dari desain proyek sederhana, dan mulai bekerja dalam modul yang kecil. Setelah pekerjaan dalam satu modul selesai dalam kurun waktu tertentu, proyek dievaluasi dan diuji. Metodologi ini memungkinkan untuk menemukan *bug* dan saran dari pelanggan sebagai bahan untuk *design sprint* berikutnya.

Metodologi *waterfall* cocok digunakan saat sebuah proyek yang memiliki gambaran produk final yang sudah jelas, saat klien tidak dapat mengubah cakupan proyek yang sudah ditentukan

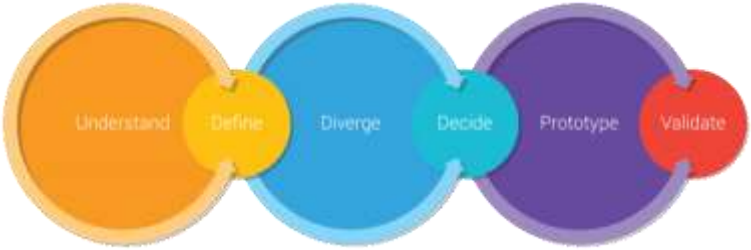
di awal, dan saat fokus utama bukanlah durasi pengerjaan namun kualitas produk. Sedangkan metodologi *agile* cocok digunakan saat durasi pengerjaan yang cepat lebih diprioritaskan kualitas produk, saat klien dapat mengubah cakupan proyek, saat tidak ada gambaran yang jelas tentang hasil final produk, dan untuk produk yang digunakan dalam industri yang cepat berubah standarnya. Pada penelitian tugas akhir ini, metodologi yang digunakan adalah Agile, karena durasi pengerjaan yang sangat terbatas dan tidak ada klien khusus sehingga belum ada gambaran produk final yang jelas menurut permintaan klien. Aplikasi *Sistem Pemantau Armada* akan disebarkan untuk seluruh jenis pengguna secara umum, sehingga dalam perkembangannya akan ada perubahan-perubahan fokus aplikasi dan desain agar sesuai dengan perusahaan yang menggunakan armada pengangkut barang secara umum.

Metodologi Agile terdiri dari beberapa jenis seperti Scrum dan *Google Design Sprint*, dan keduanya memiliki spesialisasi masing-masing. Dalam Scrum, terdapat 3 jabatan yakni *Product Owner*, *Scrum Master*, dan *Scrum Team*. *Product Owner* adalah orang yang memiliki visi tentang apa yang akan dibuat dan menjabarkan visi itu pada tim. Jabatan ini berfokus pada kebutuhan bisnis, dan memprioritaskan pekerjaan apa saja yang harus diselesaikan. *Scrum Master* adalah jabatan yang membantu agar tim dapat bekerja maksimal, dengan cara mengatur jadwal rapat, mengelola hambatan dan tantangan pengerjaan, dan bekerja sama dengan product owner untuk memastikan produk siap. S*crum Master* tidak dapat memberi tahu apa yang harus dikerjakan anggota namun lebih mengarah pada mengatur proses apa yang harus dikerjakan. Dan *Scrum Team* adalah tim yang terdiri dari 5-7 orang yang bekerja sama dan tidak memiliki pembagian tugas seperti *programmer*, *designer*, dan *tester*. [16] Dalam penerapannya di penelitian tugas akhir ini, Scrum kurang tepat digunakan karena jabatan tersebut tidak bisa diaplikasikan.

Google Design Sprint adalah salah satu dari metodologi *agile* yang dikembangkan oleh *Google Ventures* dan digunakan untuk menyelesaikan dan menguji coba desain aplikasi dengan estimasi waktu 2-5 hari. Untuk pengerjaan yang tidak memiliki waktu yang banyak, tahapan-tahapan dalam framework ini akan membantu tim dalam membuat tujuan bisnis yang tepat, memvalidasi asumsi, dan memutuskan *roadmap* produk sebelum tahap membangun aplikasi. Design sprint cocok digunakan dalam:

- Membuat produk atau service baru
- Memperbaiki desain UI/UX dari *Minimum Viable Product*
- Menambah fitur atau fungsi baru pada produk

Design Sprint memiliki 6 tahapan yakni: [17]



Gambar 2.8 Tahapan google design sprint

1. **Understand** (Apa yang *user* butuhkan, bisnis perusahaan butuhkan, dan kapasitas teknologi?)
   Metode ini dipecah dalam beberapa tahapan yakni:
   - **Presentasi kilat** untuk menjelaskan pada tim masalah dari beberapa sudut pandang yang berbeda. Diskusi ini termasuk juga mengetahui tolok ukur keberhasilan, kapasitas, dan

mendiskusikan produk yang mirip dengan proyek sebagai batu loncatan pengerjaan
- **Wawancara *user***, karena *user* adalah hakim mutlak untuk menentukan apakah sebuah produk baik atau tidak. Untuk, perlu melakukan wawancara dengan lingkup pertanyaan mengenai produk serupa, atau apa yang dibutuhkan *user* untuk menyelesaikan permasalahannya
- **Observasi langsung**, karena terkadang wawancara tidak bisa memberikan jawaban yang jelas mengenai apa yang *user* butuhkan
- **Memetakan stakeholder**, karena produk atau layanan sering memiliki beberapa tipe pengguna. Tahapan ini akan mendaftakan siapa saja orang yang masuk dalam ranah proyek
- **Menyimpulkan hasil pembelajaran**

2. **Define** (Apa strategi kunci dan fokus masalah?)
   Metode ini dipecah dalam beberapa tahapan yakni:
   - **Membuat peta** yang mendaftarkan semua tahapan dari seseorang mulai dari mempelajari produk sampai menjadi *user* yang ahli
   - ***List* prinsip desain** yang menjadi fokusan tim dalam mengerjakan produk dalam setidaknya 3 kata kunci. Pada akhirnya saat uji coba, *user* dapat menyimpulkan 3 kata kunci tersebut.
   - Bayangkan saat akan meluncurkan aplikasi, apa **kata-kata untuk mendeskripsikan produk**?
3. **Diverge** (Bagaimana pengembang dapat mencari ide sebanyak mungkin?)
   Metode ini dipecah menjadi beberapa tahapan yakni:
   - **Membuat 8 ide dalam 5 menit**. Tahapan ini sebagai pemanasan membuat sketsa aplikasi
   - **Membuat 1 ide besar dalam 5 menit**. Tahapan ini membuat ide besar aplikasi yang lebih detail lagi
   - **Membuat storyboard dari 1 ide besar**. Terkadang, ide menjadi terlalu kompleks apabila

diekspresikan dalam 1 halaman. Buatlah storyboard dari semua langkah kunci.

4. **Decide** (Memilih satu ide yang terbaik)
   Metode ini dipecah menjadi beberapa tahapan yakni:
   - **Zen Voting**. Setelah melakukan sketsa, bagikan hasilnya pada teman setim, lalu setiap orang dalam tim melakukan voting dan mereview ide.
   - **Team review**. Pada tahap ini, tim mendiskusikan ide yang terbaik dan memilih satu ide terbaik untuk dijadikan *prototype*
5. ***Prototype*** (membuat artefak yang memungkinan *user* untuk menguji coba ide)
   Prototype dapat dibedakan menjadi 4 yakni:
   - Mock, yakni membuat tiruan aplikasi baik dilakukan di Power point atau alat prototyping lainnya
   - Demo, yakni memberikan uraian fungsi-fungsi dari aplikasi
   - Video, yakni membuat *prototype* dalam bentuk video penjelasan setiap fungsi aplikasi
   - Physical *prototype*, yakni *prototype* dalam bentuk nyata, misalnya paper prototyping
6. **Validate** (Menguji ide tersebut pada *user*, stakeholder bisnis, dan para ahli lain)
   Metode ini dipecah menjadi beberapa tahapan yakni:
   - *User* Testing, untuk mendapatkan wawasan *user* tentang aplikasi
   - Stakeholder validation, untuk mendapatkan persetujuan stakeholder apakah aplikasi sudah sesuai
   - Validasi kelayakan teknis, untuk memastikan apakh desain mampu dikerjakan sesuai dengan kapasitas tim

2.2.6 Usability Testing

Sesuatu dikatakan Usable apabila orang menggunakannya tanpa rasa frustrasi. Suatu produk atau layanan dikatakan

usable apabila *user* dapat menggunakannya dengan cara yang mereka ekspektasikan tanpa ada halangan, keraguan, dan pertanyaan. [18] Usability testing dilakukan untuk mengetahui apakah orang-orang yang menggunakan produk dapat menggunakannya dengan efisien, mengetahui bagian mana yang membuat mereka bingung dan mengetahui apa yang harus dilakukan untuk mencegahnya. Testing ini dilakukan tidak setelah kita mendapatkan ide harus membuat apa, namun setelah kita memiliki gambaran untuk diuji coba. [19]

Dalam buku yang ditulis oleh Preece, ada 4 metode dalam melakukan usability, yakni: [20]

1. *Expert evaluation* atau dikenal sebagai *heuristic evaluation*, dimana pengujian dilakukan oleh orang yang berpengalaman dalam desain UI sehingga bisa meramalkan potensi masalah dari *user* yang belum pengalaman (calon pengguna aplikasi). Dengan metode ini pakar memberikan solusi dari masalah yang teridentifikasi. Para pakar yang menguji coba aplikasi tidak boleh terlibat dalam mendesain *prototype* sebelumnya. Tugas yang dilakukan dan materi yang diberikan para ahli harus mewakili calon pengguna akhir. Teknik yang bisa dilakukan untuk mengumpulkan data dalam metode ini adalah dengan *walk-through*, dan kuisioner. [21]
2. *Observational evaluation*, yakni mengumpulkan data yang memberikan informasi tentang apa yang pengguna lakukan ketika berinteraksi dengan software. Ada 2 kategori data yang dapat diperoleh yakni bagaimana menangani tugas-tugas yang diberikan, apa kesulitas, dan apa yang bisa dilakukan; dan ukuran kinerja seperti frekuensi tugas yang selesai dengan benar, waktu penyelesaian tugas, dan frekuensi kesalahannya. Teknik yang bisa dilakukan untuk mengumpulkan data dalam metode ini adalah dengan

observasi langsung, rekaman video, *software logging*, dan *verbal protocol*. [21]

3. *Survey* dilakukan untuk mengetahui pendapat pengguna akhir dan memahami keinginan mereka terhadap produk yang ada. Teknik yang bisa dilakukan untuk mengumpulkan data dalam metode ini adalah dengan wawancara dan kuisioner. [21]
4. *Experimental evaluation*, dimana evaluator memanipulasi sejumlah faktor yang terkait dengan interface aplikasi dan mempelajari efek manipulasi tersebut terhadap kinerja pengguna. Hal ini diperlukan untuk merencanakan sesuatu dengan hati-hati: pengalaman pengguna yang dibutuhkan, hipotesis yang akan diuji, strutktur tugas, waktu yang dibutuhkan untuk menyelesaikan eksperimen, dll. Teknik yang bisa dilakukan untuk mengumpulkan data dalam metode ini adalah dengan *software logging*, kuisioner, dan wawancara. [21]

Terkait beberapa teknik yang digunakan dalam keempat metode diatas, tabel 2.15 berikut adalah kelebihan dan kekurangan dari tiap teknik: [22]

Tabel 2.15 Implementasi metodologi Usability Testing

Teknik	Penjelasan	Kelebihan	Kekurangan
Kuisioner	Mendapatkan opini dan sikap pengguna terhadap kegunaan aplikasi yang dikumpulkan melalui	Cepat, murah, mudah dianalisa, anonimus	Tidak bisa memakai pertanyaan klarifikasi, tidak fleksibel, dan sulit melakukan penyelidikan

	pertanyaan-pertanyaan		
Observasi	Kebisaaan pengguna diobservasi melalui keseluruhan proses pengujian	Mendapatkan kebisaaan *user*, bukan apa yang dia pikirkan. Hasilnya dapat dibandingkan dengan metode pengumupulan data lain untuk validasi	Tidak dapat menemukan apa yang sebenarnya pengguna pikirkan atau alasan dibalik kebisaaan yang dilakukan *user*
Rekaman video	Data interaksi *user* mengenai proses usability testing direkam dengan video	Dapat melihat tindakan dari pengguna, dan juga ekspresi wajahnya	Sulit dalam menganalisis data, sulit mendapatkan sudut kamera untuk pengambilan view terbaik
Software loggding	Setiap tindakan *user* direkam melalui data logging software aplication selama proses usability testing	Tidak ada rintangan bagi *user*, metode ini dapat menginvestigasi tindakan pengguna dalam lingkungan nyata	Menggunakan software spesial, tidak dapat mengumpulkan informasi ekspresi *user*
Verbal protocol	Hasil pemikiran	Dapat mengungkapk	Tidak natural dan

	user dikumpulkan melalui tindakan dan tanggapan yang diucapkan selama proses usability testing	an alasan dibalik tindakan *user*, mengumpulkan informasi performa *user* secara simultan, dan murah	menggangu *user* dalam mengerjakan tugasnya
Wawancara	Laporan *user* mengenai aplikasi didapatkan melalui wawancara setelah proses usability testing	Level pertanyaan dapat diubah sesuai dengan konteksnya, hal yang menarik dapat lebih diinvestigasi	Hasil analisa menjadi masalah, karena perbedaan gaya dan sifat pewawancara berpengaruh pada respon partisipan

BAB III
METODOLOGI

Pada bab ini akan dijelaskan tentang metodologi sebagi panduan yang sistematis dan terarah dalam pengerjaan tugas akhir dalam proses pengerjaan tugas akhir agar bisa diselesaikan sesuai dengan batasan waktu yang disediakan oleh Jurusan Sistem Informasi ITS.

3.1 Tahapan Pelaksanaan Tugas Akhir

Tahapan-tahapan pelaksanaan yang dilakukan selama tempo waktu yang diberikan diuraikan dalam gambar 3.1.

1. **Studi Literatur**
   Pada tahapan ini dilakukan pengumpulan literatur terkait penelitian, baik berupa penelitian sebelumnya, artikel dan jurnal baik dari buku maupun *website*. Studi literatur ini dilaksanakan dalam waktu 1 minggu. Keluaran dari tahap ini adalah pemahaman konsep penelitian.

2. **Definisi *User* dan Penggalian Kebutuhannya**
   Setelah konsep penelitian dipahami, maka pengguna-pengguna aplikasi nantinya didefinisikan, baik perannya, batasan informasi yang dapat diketahui, kemampuannya menggunakan aplikasi, dan perkiraan kemampuan perangkat *Android* yang dimiliki. Setelah itu digali kebutuhan tiap jenis penggunanya melalui wawancara ahli untuk mendapatkan tanggapan yang lebih lengkap dan majemuk dari tiap narasumbernya, namun wawancara hanya bisa dilakukan pada ahli karena penelitian ini tidak memiliki studi kasus perusahaan nyata sehingga wawancara *user* tidak bisa dilakukan. Pengerjaan tahap ini diberikan alokasi waktu 1 minggu. Keluaran dari tahap ini

adalah terkumpulnya hasil wawancara berupa kebutuhan para *user* untuk dijadikan aplikasi.

3. Analisa Hasil Penggalian Kebutuhan

Pada tahapan ini, hasil wawancara beberapa ahli tentang penggalian kebutuhan tiap jenis *user* dari tahap kedua dianalisis sehingga dapat dirangkumkan apa sebenarnya kebutuhan dan batasan-batasan yang harus terkandung dalam aplikasi yang akan dirancang. Rangkuman ini akan menjadi tolok ukur dalam merancang desain antar muka dan *database* aplikasi. Pengerjaan tahap ini direncanakan menghabiskan waktu 2 minggu bersamaan dengan proses penggalian kebutuhan.

4. Design Sprint

Perancangan *blue print* aplikasi dilakukan dengan *Google Design Sprint* sehingga perancangan aplikasi nantinya terstruktur, sesuai dengan hasil analisa, dan membutuhkan waktu yang singkat. Design sprint memiliki 5 tahapan pengerjaan yakni:

A. **Define** (menentukan fokus masalah), yakni menggunakan input dari hasil analisa pengkategorian *user* dan kebutuhannya, tahap ini akan menghasilkan fokus permasalahan proses bisnis yang akan diselesaikan
B. **Diverge** (membuat beberapa ide pengembangan produk), yakni menggunakan input dari hasil define untuk menghasilkan beberapa ide pengembangan produk yang cocok
C. **Decide** (Menentukan satu ide untuk dikembangkan), yakni memilih satu yang terbaik dari beberapa ide tahap diverge untuk dibuat *prototype*-nya
D. Pembuatan ***prototype***, yakni membuat *prototype* untuk mendetailkan ide penyelesaian. *Prototype* inilah yang nantinya akan diujikan ke *user*
E. **Validasi**, merupakan tahap terakhir design sprint untuk mendapatkan tanggapan dari stakeholder, *user*,

dan para ahli mengenai kecocokan desain aplikasi dengan kebutuhan mereka yang sesungguhnya

Pengerjaan tahap ini direncanakan menghabiskan waktu 2 minggu dengan output berupa *prototype* aplikasi.

5. Pengembangan Aplikasi

Tahapan ini adalah tahapan inti dimana semua aplikasi sesuai *blue print* dirancang sampai bisa digunakan oleh *end-user* baik dari segi *user interface*, fungsi-fungsi, dan *database*-nya. Output dari tahapan ini adalah *end-user application*. Tahapan ini direncanakan memakan waktu 9 minggu.

6. Usability Testing

Pada tahap ini aplikasi yang telah dirancang sebelumnya diuji pada beberapa *user* secara random yang memiliki pengetahuan tentang desain namun sama sekali tidak mengerti tentang desain aplikasi penelitian ini karena calon *user* yang menggunakan sebelumnya juga tidak tahu rancangan aplikasi. *Usability testing* ini dilakukan untuk mengetahui apakah *user* dapat menggunakan aplikasi sehingga diharapkan dapat memberikan gambaran mana *User interface* (UI) dan *User* Experience (UX) yang masih kurang, mana fungsi yang belum bekerja maksimal, dan melihat *bug* aplikasi yang tidak terdeteksi saat pengembangannya. Output dari tahapan ini adalah dokumen review *usability testing* yang menjadi acuan dalam pemutakhiran aplikasi sebelum dipublikasikan. *Usability testing* direncakan akan memakan 2 minggu.

7. Pemutahiran Aplikasi

Tahapan terakhir ini adalah perbaikan aplikasi sampai menjadi layak untuk dipublikasikan kepada *user* berdasarkan dokumen *review usability testing* yang didapat pada tahap sebelumnya. Output dari tahapan ini adalah *final end-user application* untuk dipublikasikan kepada *user* yang menjadi target dan dokumen *source code*

sebagai pembelajaran untuk penelitian selanjutnya. Pengerjaan tahapan ini menghabiskan waktu 2 minggu, dimana tahap ini sudah dimulai diminggu *usability testing* masih berjalan karena hasil usability testing pada minggu pertama sudah bisa dijadikan bahan pengerjaan.

8. Penyusunan Buku Tugas Akhir

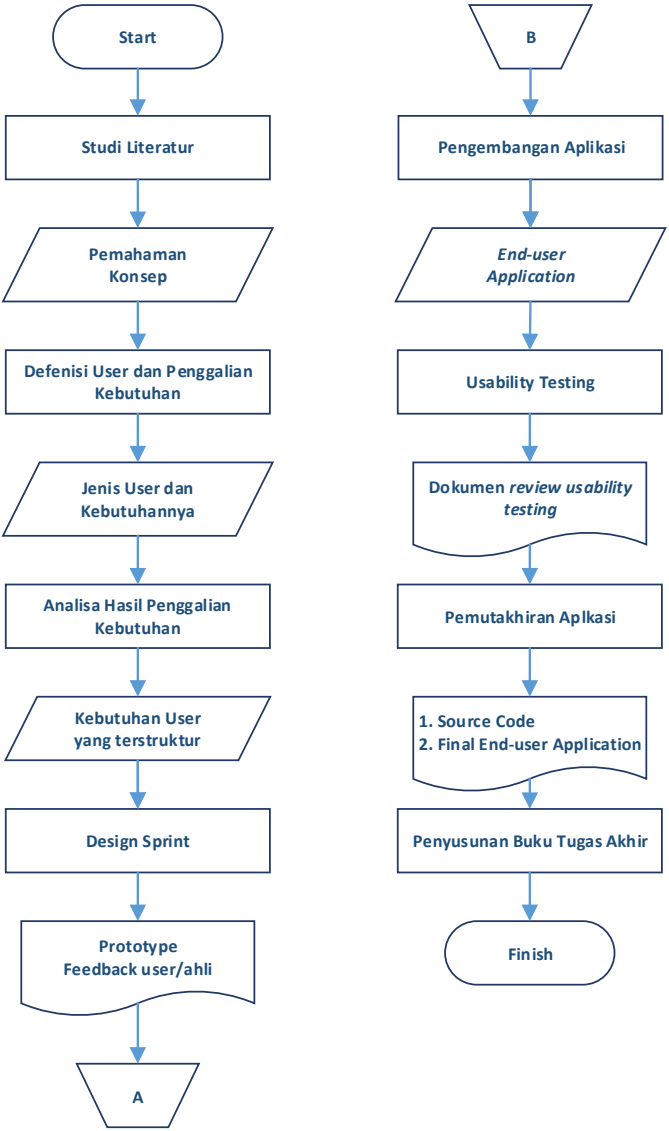
Aktivitas ini adalah pendukung dari penelitian Tugas Akhir untuk mendokumentasikan keseluruhan tahap pengerjaan dalam satu buku utuh. Aktivitas ini sudah mulai dilakukan sejak awal kegiatan sampai pada akhir kegiatan.

3.2 Alat dan bahan yang Digunakan

Dalam pengerjaan Tugas Akhir ini, alat dan bahan yang digunakan adalah

A. *Android* Studio sebagai IDE dalam perancangan aplikasi yang berbasis *Android*
B. *Firebase* API sebagai dasar dalam membuat *realtime database*
C. GoogleMap API untuk membantu dalam menampilkan peta rute perjalanan dan menyarankan rute yang tercepat menurut algoritma GoogleMap

   Daftar barang yang dijual di swalayan sebagai contoh dalam membuat *database* barang.



Gambar 3.1 Tahapan pelaksanaan tugas akhir

(Halaman Sengaja Dikosongkan)

BAB 4
PERANCANGAN

Pada bab ini akan dijelaskan tentang gambaran umum sistem, analisis dan desain sistem sesuai dengan metodologi Agile *Google Design Sprint*.

4.1 Gambaran Umum Sistem

Konsep arsitektur aplikasi sistem pemantau armada dapat dilhat dari gambar 2.2 dan konsep proses bisnisnya dapat dilihat dari gambar 2.1. Pada gambar tersebut dijelaskan bahwa sistem dimulai saat manajemen perusahaan membuat penugasan baru pada *form* yang ada di aplikasi manajemen perusahaan yang berisi pelanggan tujuan, barang pesanan (nama dan jumlah), alamat pelanggan, dan token pesanan. Pada saat penugasan ini, pihak manajemen juga mengecek armada yang tersedia untuk diberikan tugas. Setelah mengisi penugasan dan memilih armada, penugasan dikirim ke aplikasi armada tujuan. Proses bisnis kemudian dilanjutkan di aplikasi sopir.

Saat mengkonfirmasi pengerjaan tugas yang diterima sopir, artinya sopir berada dalam status sedang mengerjakan tugas. Dalam hal ini sopir akan secara simultan mengirimkan data lokasinya ke *server* agar bisa dipantau oleh pihak manajemen. Setelah sampai di pelanggan tujuan, sopir memberikan aplikasinya kepada pelanggan agar pelanggan dapat mengkonfirmasi barang yang sampai kepadanya. Konfirmasi ini akan dikirimkan ke pihak manajemen.

Sistem kemudian melakukan perhitungan barang yang sudah sampai dengan barang yang yang seharusnya diantarkan. Apabila barang sampai sudah sesuai, maka penugasan pada sopir diberi status selesai. Namun apabila tidak sesuai, maka diberikan status tidak selesai. Manajemen perusahaan akan

mengecek apakah penugasan ada yang belum selesai. Jika belum selesai, manajemen perusahaan akan memilih apakah tetap dilanjutkan dengan sopir yang sama atau dengan sopir lain. Tugas yang belum selesai akan dibuat menjadi tugas baru dengan label "*rework task*".

Dari proses bisnis tersebut, dapat disimpulkan bahwa sistem pemantau armada ini membutuhkan 3 produk. Yang pertama adalah aplikasi untuk digunakan manajemen perusahaan, kemudian yang kedua adalah aplikasi yang digunakan sopir armada, dan yang ketiga adalah *web-based application* yang memungkinkan manajemen perusahaan memantau keseluruhan armada dengan tampilan yang lebar (*desktop*) karena tampilan dari *smartphone* kecil dan sulit untuk melihat seluruh armada yang tersebar.

Konsep umum dari arsitektur dari sistem pemantau armada ini dapat dilihat pada gambar 2.2. Untuk dapat beroperasi optimal, sistem ini harus memiliki 3 jenis produk, yakni:

1. Aplikasi berbasis *Android* untuk digunakan oleh manajemen perusahaan. Dengan aplikasi ini, pihak manajemen dimungkinkan untuk dapat menambahkan dan mengubah data dalaam daftar armada yang dimilikinya, menambahkan penugasan baru pada armada yang terdaftar, mengalihkan penugasan dari satu armada ke armada lain, mengkases informasi lokasi armada secara sederhana, dan menerima konfirmasi penerimaan barang.
2. Aplikasi berbasis *web* untuk digunakan oleh manajemen perusahaan. Dengan aplikasi ini, pihak manajemen dimungkinkan untuk mengakses lokasi keseluruhan armada yang dimilikinya secara *realtime*.

Aplikasi ini terbatas hanya untuk memantau lokasi armada saja, karena fungsi-fungsi penugaan dilakukan di aplikasi *Android* milik manajemen perusahaan.

3. Aplikasi berbasis *Android* untuk digunakan oleh sopir armada. Dengan aplikasi ini, sopir-sopir armada dimungkinkan untuk mengakses penugasan yang diterimanya, mengirimkan riwayat lokasinya secara *realtime* saat mengeksekusi tugas, mengkonfirmasi barang yang dikirimkan ke pelanggan/penyelesaian tugas, dan memperbaharui profil data miliknya. Pengiriman riwayat lokasi ke *server* dilakukan jika sopir dalam keadaan bekerja/mengirimkan barang. Sehingga saat tidak mengirimkan barang, perusahaan menanggapi status armada dengan *idling*/menganggur.'

Ketiga aplikasi tidak memiliki *database* lokal untuk menyimpan data baik sementara ataupun permanen. Semua data tersimpan secara cloud menggunakan fitur *Realtime Database* dari *Google Firebase*. Pertukaran dan pengiriman data dari ketiga aplikasi ke *server* dan dari *server* ke aplikasi menggunakan *library* org.*JSON*. Dengan *firebase*, data bisa dapat tersimpan sementara di perangkat *Android* saat tidak teknoneksi dengan internet, dan akan langsung mengirimkannya ke *server* saat koneksi internet sudah stabil.

4.2 " *How Might We..*" Notes

"*How Might We*" adalah digunakan dalam fase memahami proyek. Pada dasarnya HMW melibatkan anggota tim untuk menuliskan pemikiran mereka dimana setiap ide dituliskan dalam satu sticky notes. Tiap ide dituliskan dalam bentuk pernyataan dan diawali "*How Might We*". Selama *Google Design Sprint*, kita tidak perlu menghabiskan banyak waktu

untuk membuat dan menyempurnakan pernyataan HMW, tetapi mengasilkan sebanyak mungkin HMW. [23]

Dalam penelitian tugas akhir ini, HMW dibuat dan dikelompokkan menjadi 3 berdasarkan 3 produk utama dari penelitian ini, seperti yang bisa dilihat pada table 4.1 dibawah ini:

Tabel 4.1 "How Might We"

Produk	*How Might We..*
Smartphone Manajemen Perusahaan	1. Menggunakan API *Firebase* agar terhubung dengan *realtime database* 2. Menggunakan API *Google maps* agar bisa menggunakan fasilitas peta dunia 3. Mendaftarkan dan memodifikasi akun manajemen perusahaan 4. *Login* hanya dengan menggunakan akun manejemen perusahaan 5. Mendaftarkan, memodifikasi, dan menghapus akun sopir 6. Membuat penugasan baru untuk satu sopir 7. Membuat Token sebagai verifikasi penyelesaian tugas setiap sopir 8. Mengecek lokasi terkini setiap sopir terdaftar secara visual dalam peta dengan data dari *realtime database* 9. Mengecek *activity* terkini setiap armada (*ignition*) 10. Memberikan status penugasan selesai untuk sopir yang sudah menyelesaikan tugasnya 11. Memberikan status penugasan selesai untuk sopir yang belum menyelesaikan tugas berdasarkan

	verifikasi pelanggan namun tidak boleh melanjutkannya lagi 12. Membuat penugasan ulang untuk supir yang tidak dapat menyelesaikan tugasnya berdasarkan verifikasi pelanggan
Smartphone Sopir	1. Menggunakan API *Firebase* agar terhubung dengan *realtime database* 2. Menggunakan API *Google maps* agar bisa menggunakan fasilitas peta dunia 3. Memodifikasi data akun sopir 4. Mengirimkan lokasi terkini ke manajemen perusahaan melalui *realtime database* 5. Memberikan konfirmasi untuk mulai mengerjakan tugas 6. Memberikan konfirmasi bahwa armada sedang menganggur 7. Mengirimkan hasil penyelesaian tugas ke *server* menggunakan token 8. Melihat *list* penugasan yang harus dikerjakan sopir 9. Melihat *list* penugasan yang telah dikerjakan sopir
Desktop Manajemen Perusahaan	1. Menggunakan API *Firebase* agar terhubung dengan *realtime database* 2. Menggunakan API *Google maps* agar bisa menggunakan fasilitas peta dunia 3. *Login* hanya dengan menggunakan Akun manejemen perusahaan 4. Mengecek lokasi terkini setiap sopir secara visual 5. Mengecek *activity* terkni setiap armada

4.3 *Lightning Demos*

Lightning demos adalah melihat solusi yang sudah berhasil untuk kasus serupa. Tahapan ini mengamati ide-ide tersebut baik dengan observasi maupun studi literatur. Beberapa produk yang dianalisis pada *lightning demos* ini adalah hasil studi literatur karena merupakan aplikasi berbayar dan tidak memerlukan interaksi langsung pada aplikasi, seperti:

4.3.1 Trackimo

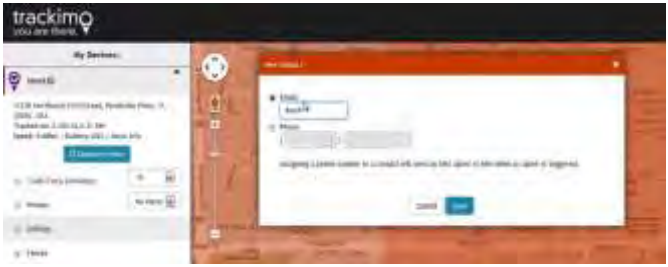
Sebagai *Fleet tracking system*, Trackimo memiliki fitur dan *User interface* sebagai berikut:

1. Perangkat keras Trackimo

Gambar 4.1 Trackimo - Perangkat keras

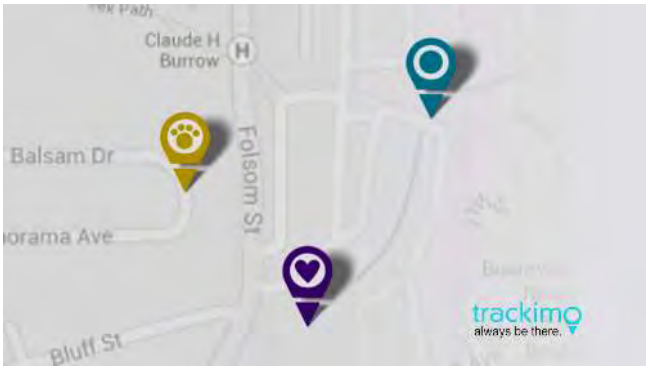
Trackimo bekerja dengan melacak alat khusus yang dibawa-bawa oleh orang yang ingin dilacak. Bentuk alatnya dapat dilihat seperti gambar 4.1 diatas.

2. Mendaftar alat baru
   Satu alat trackimo baru yang ingin dilacak didaftarkan terlebih dahulu ke dalam akun pemilik, lalu mengisi data dan pengaturan pelacakannya. Pengaturan dapat dilihat pada gambar 4.2.



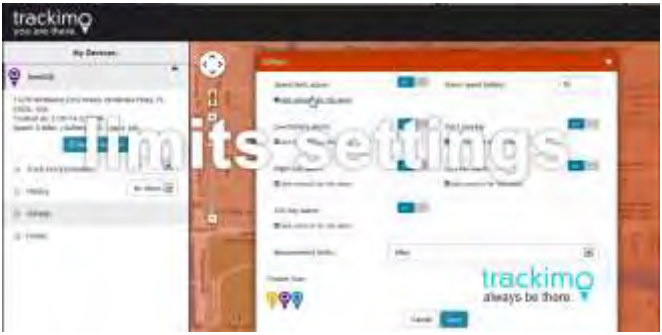
Gambar 4.2 Trackimo – Mendaftar alat baru

3. Visualisasi lokasi tiap perangkat Trackimo
   Beberapa alat Trackimo yang menyebar dapat divisualisasikan dengan berbeda-beda simbol dalam 1 peta seperti pada gambar 4.3.



Gambar 4.3 Trackimo - Visualisasi lokasi tiap perangkat

4. Pengaturan pembatasan kecepatan
   Trackimo memiliki fitur untuk memberikan batasan kecepatan, dimana apabila melebihi kecepatan tersebut, alat akan mengirimkan peringatan ke akun pemiliknya. Pengaturan tersebut dapat dilihat pada gambar 4.4.



Gambar 4.4 Trackimo - Pengaturan pembatasan kecepatan

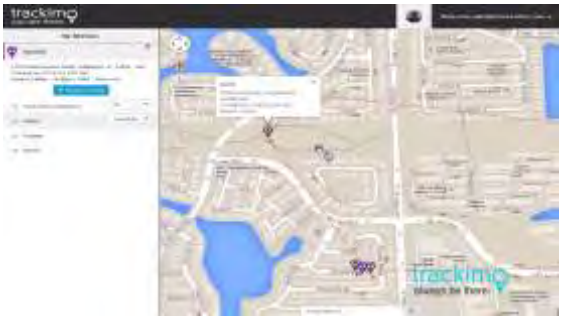
5. Pengaturan pembatasan area
   Trakimo juga memiliki fitur untuk memberikan batasan area, yang akan mengirimkan peringatan ke akun pemilik apabila melewati zona. Pengaturan tersebut dapat dilihat pada gambar 4.5.

Gambar 4.5 Trackimo – Pengaturan pembatasan area

6. Fitur SOS Trackimo
   Perangkat Trackimo memiliki tombol SOS seperti pada gambar 4.6, yang berfungsi apabila seseorang tersesat dan ingin dijemput. Fitur ini biasanya digunakan oleh

orang tua. Sehingga pemilik akun dapat mengerti dimana lokasi alat dan bisa menjemput ke lokasi tersebut. Lokasi tersebut divisualisasikan dalam peta seperti pada gambar 4.7.

Gambar 4.6 Trackimo - Perangkat pengaktif SOS

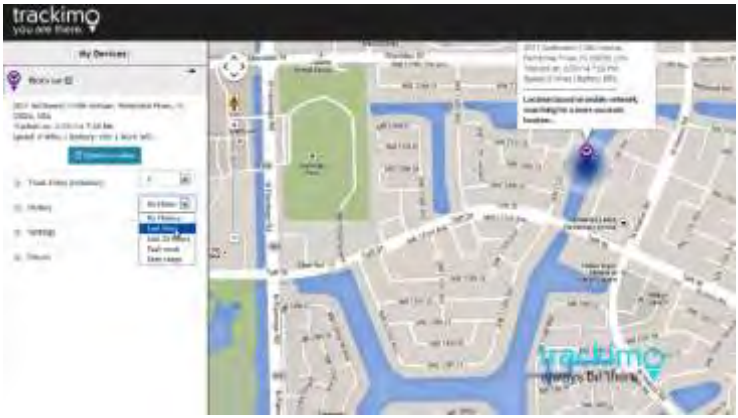


Gambar 4.7 Trackimo - UI fitur SOS

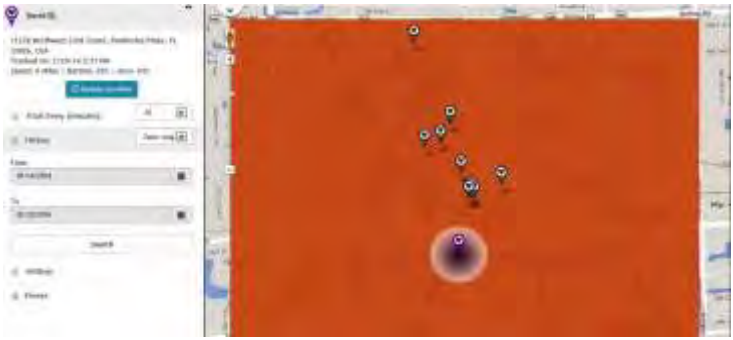
7. Fitur Riwayat perjalanan perangkat

Trackimo memiliki fitur untuk melihat riwayat perjalanan tiap perangkat yang dikelompokkan berdasarkan waktu. Namun melihat riwayat lokasi tidak bisa dilakukan untuk lebih dari 1 alat sekaligus. Daftar riwayat dapat dipilih seperti pada gambar 4.8.

Kemudian pada gambar 4.9 ditampilkan riwayat lokasi alat dari tanggal 15-20 Februari 2014.



Gambar 4.8 Trackimo - Melihat daftar riwayat



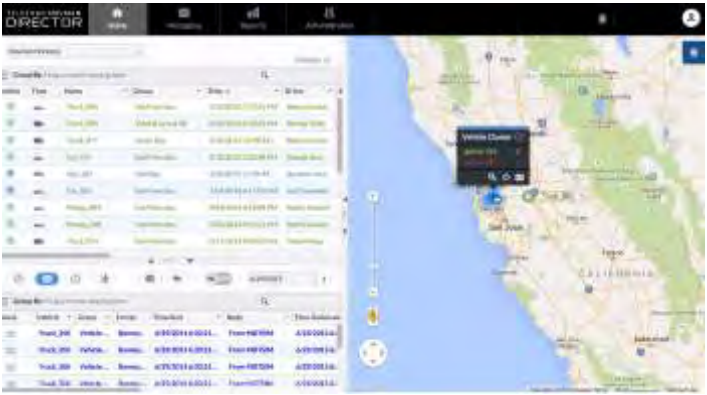
Gambar 4.9 Trackimo - Melihat riwayat tersimpan

4.3.2 Teletrac Navman

Untuk menggunakan fitur *administrator* yang bisa me*monitor* seluruh armadanya, Teletrac Navman memiliki Aplikasi

bernama Teletrac Navman Director, dengan Fitur dan *user interface* sebagai berikut:

1. Pengelompokan Kendaraan
   Teletrac Navman mengelompokkan kendaraan-kendaraan berdasarkan lokasi. Pada gambar 4.10 dapat dilihat bahwa di Daerah Palo Alto terdapat 5 kendaraan, dimana 3 diantaranya dalam keadaan menyala.



Gambar 4.10 Teletrac Navman – Pengelompokan kendaraan

   Teletrac Navman mengelompokkan kendaraan-kendaraan berdasarkan lokasi. Dari gambar 4.10 diatas dapat dilihat bahwa di Daerah Palo Alto terdapat 5 kendaraan, dimana 3 diantaranya dalam keadaan menyala.

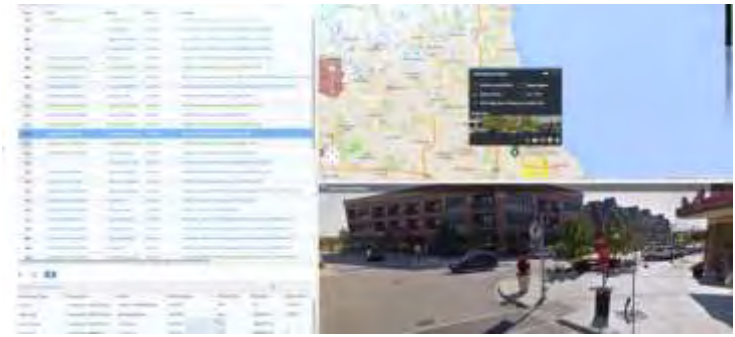
2. Detail Kendaraan
   Detail setiap kendaraan bisa dilihat dengan meletakkan kursor pada posisi kendaaraan, lalu aplikasi akan menampilkan Waktu beroperasi terakhir, pengemudi,

kontak pengemudi, lokasi kendaraan, total jarak tempuh, total jam kendaraan menyala, Status *ignition* mesin, dan kecepatan kendaraan saat ini. Detail kendaraan dapat dilihat pada gambar 4.11.

Gambar 4.11 Teletrac Navman - Detail kendaraan

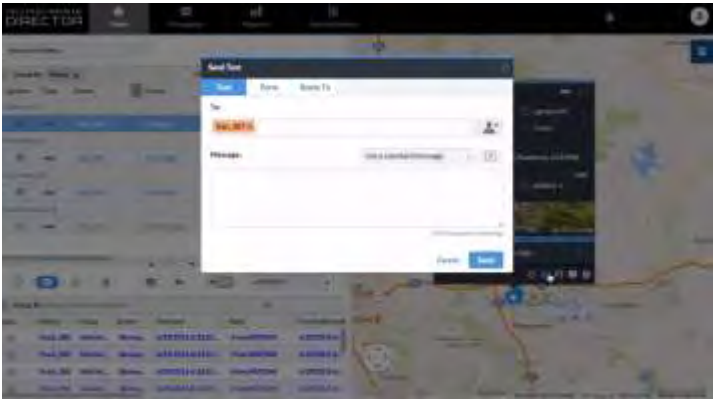
3. *Street view* di lokasi kendaraan
   Teletrac Navman memanfaatkan fasilitas *Google maps* dan *Google Street View* untuk melihat lokasi kendaraan secara detail seperti pada gambar 4.12 dan gambar 4.13.

Gambar 4.12 Teletrac Navman - Street View 1

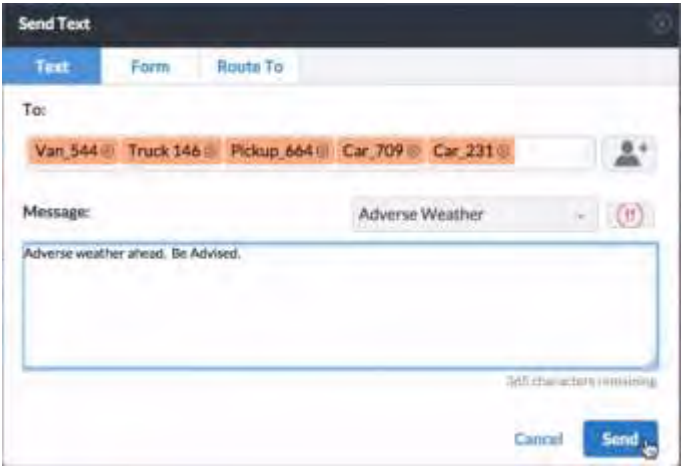


Gambar 4.13 Teletrac Navman – Street View 2

4. Mengirimkan pesan biasa pada kendaraan
   Aplikasi ini memiliki fitur untuk mengirimkan pesan biasa baik untuk 1 kendaraan maupun banyak kendaraan sekaligus seperti gambar 4.14 dan 4-15 dibawah.



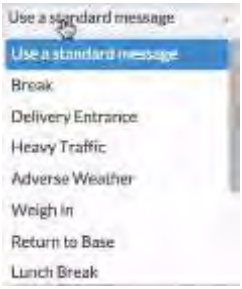
Gambar 4.14 Teletrac Navman - Mengirimkan pesan pada 1 kendaraan



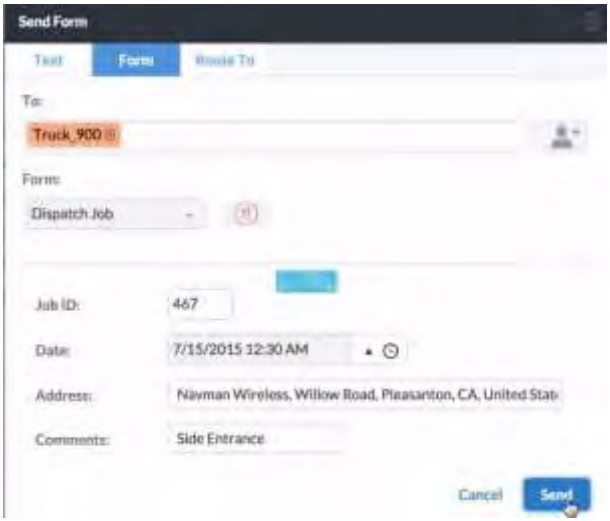
Gambar 4.15 Teletrac Navman - Mengirimkan pesan pada >1 Kendaraan

5. Mengirimkan pesan perintah pada kendaraan
   Pesan standar seperti pada gambar 4.16 merupakan pesan *template* yang sudah disediakan sehingga memudahkan penulisan pesan pada pengemudi

kendaraan. Sementara pesan perintah untuk armada ditampilkan pada gambar 4.17.



Gambar 4.16 Teletrac Navman - Pesan standar

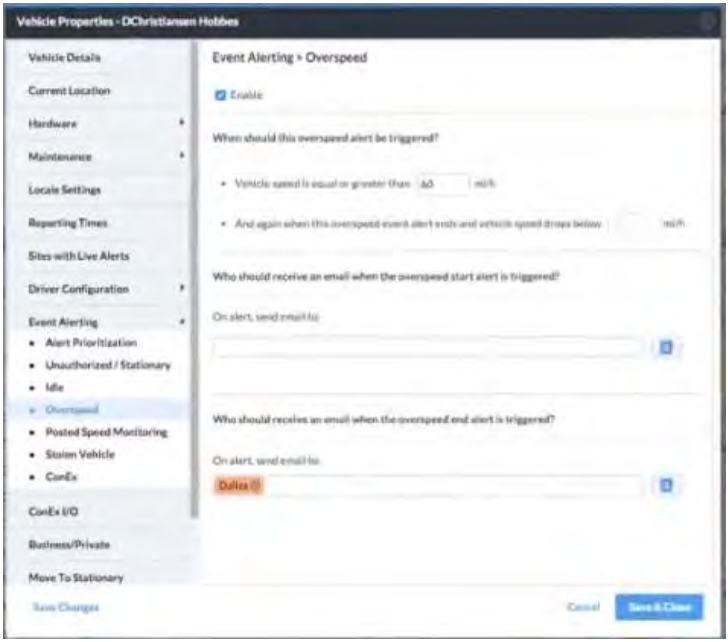


Gambar 4.17 Teletrac Navman - Membuat pesan perintah penugasan

Menggunakan Tab *Form* pada gambar 4.17, pihak manajemen dapat membuat pesan perintah untuk

pengemudi dengan mudah, hanya dengan mengisikan *template*.

6. Perilaku pengemudi
   Pihak manajemen dapat membuat batasan kecepatan yang diperbolehkan. Apabila melebihi, maka akan diberikan peringatan. Pengaturannya dapat dilihat seperti gambar 4.18.



Gambar 4.18 Teletrac Navman - Pengaturan perilaku pengemudi

7. *Driver Efficiency Report*
   Driver efficiency report adalah laporan mengenai rata-rata berhenti kendaraan, rata-rata lamanya kendaraan menganggur, penggunaan bahan bakar, dan skor

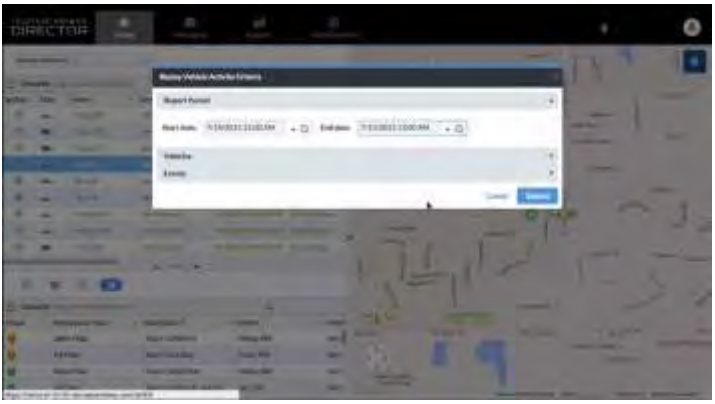
pengemudi. Laporan ini dibuat dalam bentuk grafik seperti gambar 4.19.

Gambar 4.19 Teletrac Navman - Laporan *activity* pengemudi

8. Menyetel pesan standar
   Untuk mempermudah mengirimkan pesan yang sudah *default*, Teletrac Navman memberikan fitur untuk membuat pesan standar, sehingga saat ingin mengirimkan pesan, manajemen bisa langsung memilih salah satu dari pesan standar yang tersedia.

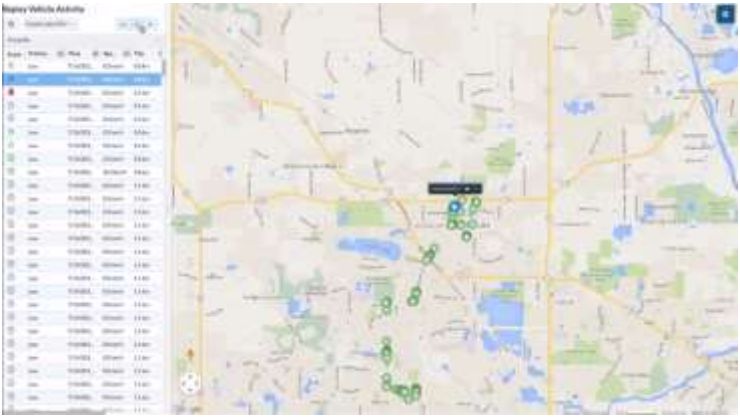
Gambar 4.20 Teletrac Navman - Membuat pesan standar

9. Riwayat rute kendaraan
   Sebelum melihat rute mana saja yang d*item*puh, pertama-tama manajemen memilih periode riwayat yang ingin ditampilkan seperti pada gambar 4.21.



Gambar 4.21 Teletrac Navman - Daftar riwayat kendaraan

Kemudian Pada gambar 4.22, dapat dilihat titik-titik lokasi yang dilengkapi dengan panah arah kendaraan (titik hijau) sampai lokasi terkininya (titik biru).



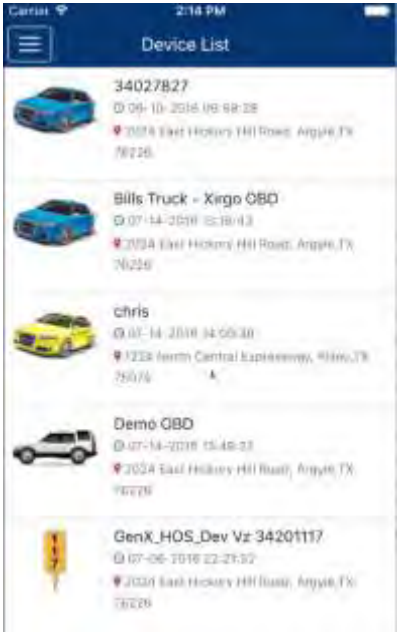
Gambar 4.22 Teletrac Navman - Riwayat perjalanan kendaraan

4.3.3 *Rhino Fleet Tracking* (RFT)

Rhino Fleet Tracking adalah aplikasi pelacak kendaraan dengan tampilan yang sangat sederhana. Berikut adalah menu-menu yang disajikan:

1. *Login*
   gambar 4.23 merupakan tampilan awal aplikasi RFT. Sebelum bisa mengakses informasi kendaraan, pertama sekali harus *login* sesuai dengan *username* dan *password* yang terdaftar.
2. Daftar kendaraan yang dimiliki akun
   Setelah berhasil *login*, aplikasi menampilkan daftar kendaraan yang dimiliki akun tersebut beserta dengan lokasi terkhirnya, seperti gambar 4.24.

Gambar 4.23 Rhino Fleet Tracking – Login



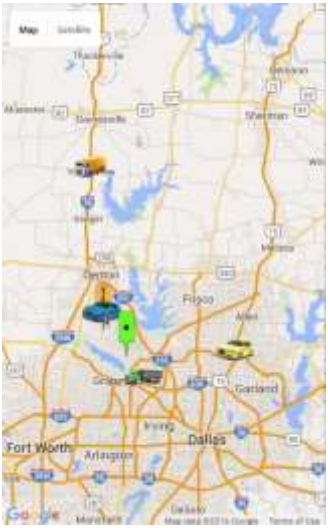
Gambar 4.24 Rhino Fleet Tracking – Daftar Kendaraan

3. Posisi terkini kendaraan

   Jika salah satu kendaraan dipilih, maka akan terlihat visualisai dalam peta lokasi kendaraan beserta kecepatan dan waktu terakhir kendaraan mengirimkan lokasinya, seperti pada gambar 4.25.

   Tidak hanya 1 kendaraan saja yang bisa dilihat lokasinya, namun seluruh kendaraan terdaftar dapat dilihat dalam 1 waktu dan 1 peta, seperti pada gambar 4.26.

Gambar 4.25 Rhino Fleet Tracking – Detail terkini 1 kendaraan



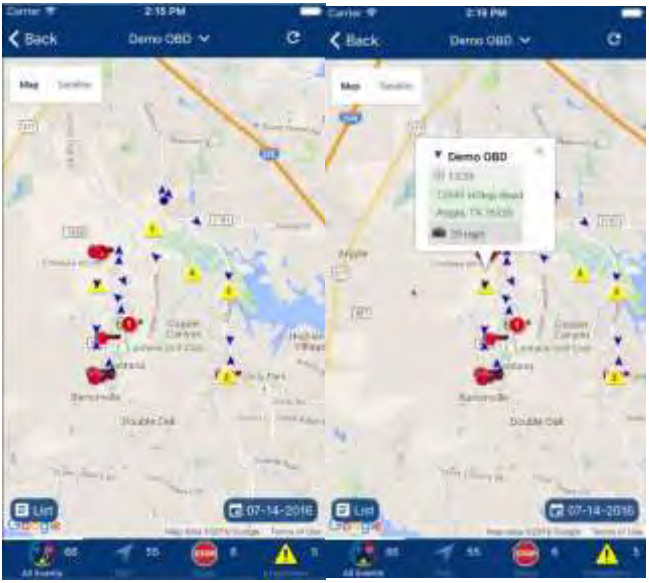
Gambar 4.26 Rhino Fleet Tracking – Posisi terkini seluruh kendaraan

4. Riwayat Kendaraan

   Untuk melihat riwayat lokasi kendaraan, pertama kali riwayat yang dikelompokkan berdasarkan tanggal, seperti yang dapat dilihat dalam gambar 4.27. Setelah itu dapat dilihat riwayat lokasinya baik menggunakan visualisasi peta seperti pada gambar 4.28 maupun dalam bentuk daftar seperti pada gambar 4.29.

   Setiap kita meletakkan mengklik salah satu ikon tersebut, maka aplikasi menjelaskan informasi ikon tersebut seperti yang dapat dilihat dalam gambar 4.28. Manajemen juga dapat melihat detail informasi dalam bentuk *list* apabila tombol *list* ditekan sehingga muncul tampilan seperti gambar 4.29.

Gambar 4.27 Rhino Fleet Tracking – Memilih riwayat kendaraan tersimpan



Gambar 4.28 Rhino Fleet Tracking – Riwayat kendaraan (Visualisasi)

Gambar 4.29 Rhino Fleet Tracking – Riwayat kendaraan (List)

4.4 Diverge

Tujuan dari tahapan ini adalah mengeksplorasi sebanyak mungkin ide yang bisa dijadikan solusi untuk produk. Ide-ide tersebut bisa realistis ataupun tidak, namun dari ide-ide yang muncul, kita memiliki wawasan mana yang lebih baik. Dengan begitu saat memilih salah satu dari banyak ide tersebut, kita sudah lebih percaya diri karena sudah memastikan tidak ada yang lebih baik dari itu. Lain hal apabila hanya memikirkan 1 ide saja, kita tidak yakin apakah ide itu sudah yang terbaik atau tidak.

4.4.1 Ide 1

4.4.1.1 *Smartphone* Manajemen Perusahaan

(Lampiran 1)

4.4.1.2 *Smartphone* Sopir Armada
(Lampiran 2)

4.4.1.3 *Desktop* Manajemen Perusahaan
(Lampiran 3)

4.4.2 Ide 2

4.4.2.1 *Smartphone* Manajemen Perusahaan
(Lampiran 4)

4.4.2.2 *Smartphone* Sopir Armada
(Lampiran 5)

4.4.2.3 *Desktop* Manajemen Perusahaan
(Lampiran 6)

4.4.3 Ide 3

4.4.3.1 *Smartphone* Manajemen Perusahaan
(Lampiran 7)

4.4.3.2 *Smartphone* Sopir Armada
(Lampiran 8)

4.4.3.3 *Desktop* Manajemen Perusahaan
(Lampiran 9)

4.5 Converge

4.5.1 Asumsi/ Tabel uji

Semua asumsi dan kemungkinan dalam sistem harus dianalisa untu meminimalisir *bug*. Setelah semua asumsi dikumpulkan,

tentukan bagaimana pengujian yang tepat untuk setiap asumi yang ada. Tabel 4-2 berikut adalah asumsi dan pengujian untuk sistem pemantau armada yang d*item*ukan:

Tabel 4.2 Tabel asumsi-uji

Asumsi	Pengujian
Kondisi *database* di tiap perangkat saat tidak terhubung dengan internet	- Mengecek kondisi di masing-masing online *Firebase*, *smartphone* manajemen perusahaan, *desktop* manajemen perusahaan, dan *smartphone* sopir
Form penerimaan barang sudah terisi, namun tidak masuk ke *database*	- Cek koneksi, kapasitas tersisa dari *Firebase*, dan *bug*
Kode token aman dari sopir	- Cek hak akses untuk melihat token
Sopir hanya bisa *login* dengan akun yang sudah dibuat perusahaan (*Username* dan *password* yang sesuai)	- *Login* dengan akun perusahaan, *password* salah, dan akun *dummy*

4.5.2 *User* Story, Skenario, dan Storyboard

Dijabarkan dalam Tabel 4-3, dengan menggunakan aplikasi *android* manajemen perusahaan, sebagai *administrator*:

Tabel 4.3 Use case android manajemen perusahaan

US-AM1-1	*User* dapat mendaftarkan akunnya pada aplikasi (*Sign up*) sehingga dapat menggunakan akses penuh aplikasi.

US-AM1-2	*User* dapat menggunakan aplikasi menggunakan akun miliknya (*sign in*) sehingga 1 akun yang telah didaftkarkan dapat dipakai di *smartphone* manapun dengan data yang sama.
US-AM1-3	*User* dapat mendaftarkan akun armada milik perusahaan, sehingga sopir dapat menggunakan aplikasi *android* sopir perusahaan.
US-AM1-4	*User* dapat mengedit informasi akun perusahaan sehingga informasi tentang perusahaan tetap *update.*
US-AM1-5	*User* dapat mengedit informasi akun armada milik perusahaan sehingga informasi tentang sopir milik perusahaan tetap *update.*
US-AM2-1	*User* dapat membuat penugasan baru sehingga sopir dapat melaksanakan pengantaran barang.
US-AM2-2	*User* dapat melihat status sopir (lokasi dan tugas) sehingga *user* bisa mengambil kebijakan mengenai pemberian tugas berdasarkan lokasi sopir.
US-AM2-3	*User* dapat membatalkan pengiriman barang sehingga transaksi yang dibatalkan tidak perlu dikirimkan.
US-AM2-4	*User* dapat melihat penugasan yang sedang berjalan sehingga *user* tahu mana tugas yang sedang dikerjakan dan mana yang belum dikerjakan.
US-AM2-5	*User* dapat melihat riwayat penugasan yang sudah berakhir sehingga tahu rute mana saja yang dilalui saat pengantaran tugas, dan berapa barang yang sampai.
US-AM3-1	*User* dapat menerima laporan barang yang sampai dari pelanggan tanpa manipulasi

	sehingga data pengiriman akurat dan pelanggan puas.
US-AM4-1	*User* dapat membuat penugasan lanjutan pada sopir yang sama sehingga tugas sopir lain tidak bertambah akibat kelalaian seorang sopir.
US-AM4-2	*User* dapat membuat penugasan lanjutan pada sopir yang berbeda sehingga penugasan dapat diselesaikan meskipun sopir yang lalai tidak memungkinkan untuk menyelesaikan tugas.
US-AM4-3	*User* dapat mengalihkan penugasan ke sopir lain sehingga sopir yang berhalangan mengantar barang dapat dialihkan tugasnya.
US-AM5-1	*User* dapat membuat, memodifikasi, dan menghapus gudang milik perusahaan sehingga sopir tahu harus mengambil barang pesanan kemana.
US-AM6-1	*User* dapat me-*reset password* apabila lupa sehingga akun bisa tetap dipakai
US-AM7-1	*User* dapat melihat kinerja tiap armada miliknya

Dijabarkan dalam Tabel 4-4, dengan menggunakan aplikasi android Sopir Armada, sebagai sopir:

Tabel 4.4 Use case android android sopir perusahaan

US-AS1-1	*User* dapat mengkases aplikasi dan data didalamnya menggunakan akun miliknya (*sign in*) sehingga 1 akun yang telah didaftkarkan dapat dipakai di *smartphone* manapun dengan data yang sama.
US-AS1-2	*User* dapat menerima dan melihat tugas yang diberikan perusahaan sehingga *user* tahu apa yang harus diantarkannya, dan kemana harus diantarkan.

US-AS1-3	*User* dapat mengajukan agar penugasan yang diberikan dialihkan ke sopir lain sehingga sopir yang berhalangan tidak dipaksa untuk menyelesaikan tugas.
US-AS2-1	Melihat tugas yang sudah diselesaikan sehingga *user* tahu tugas mana saja yang sudah selesai dan ada pemisah antara tugas yang selesai dan belum.
US-AS3-1	*User* dapat menyimpan riwayat lokasi saat melaksanakan pengiriman barang sehingga perusahaan tahu apa saja jalur yang dilalui oleh *user*.
US-AS3-2	*User* dapat mengisi *form* penyelesaian tugas dan mengirimkannya ke *server* sehingga perusahaan tahu apakah yang dikerjakannya sudah sesuai atau kurang dari target.
US-AS3-3	*User* dapat mengajukan pembatalan pengantaran beberapa tugas dalam sekali pengantaran sehingga sopir yang mendadak berhalangan dapat dialihkan ke sopir lain.
US-AS3-4	*User* dapat menerima konfirmasi tugas dilanjutkan sehingga *user* tahu kalau tugas yang belum diselesaikannya harus diselesaikan sendiri atau dialihkan ke sopir lain.
US-AS3-5	*User* dapat menerima pemberitahuan baik apabila penugasan dibatalkan maupun dija pengajuan pengalihan tugasnya disetujui sehingga pesanan dari transaksi yang dibatalkan tidak perlu diantarkan.
US-AS4-1	*User* dapat mengirimkan lokasi ke *server* secara simultan sehingga perusahaan tahu lokasinya terkini.

US-AS4-2	*User* dapat melihat lokasinya terkini, lokasi-lokasi gudang, dan lokasi perusahaan dalam visualisasi peta.
US-AS5-1	*User* dapat memodifikasi informasi akunnya sehingga informasi tentang *user* tetap *update.*
US-AS6-1	*User* dapat me-*reset password* apabila lupa sehingga akun bisa tetap dipakai

Dijabarkan dalam Tabel 4-5, dengan menggunakan website manajemen perusahaan, sebagai *administrator*:

Tabel 4.5 Website manajemen perusahaan

US-WM1-1	*User dapat login ke website* hanya bisa menggunakan akun perusahaan, sehingga sopir tidak bisa sembarangan membuka website
US-WM2-1	*User* dapat melihat lokasi seluruh armada sehingga *user* tahu apakah ada sopir yang menyimpang dari lokasi seharusnya.
US-WM2-2	*User* tahu apakah sopir sedang bekerja atau beristirahat.

4.5.2.1 *Smartphone* Manajemen Perusahaan

1. *User* membuat akun baru

   Skenario (gambar pada lampiran 10) :

   a. Perusahaan Alfasejahtera adalah sebuah *retail* yang baru saja membeli 10 truk baru untuk membantu mengirimkan barang jualannya.

   b. Alfasejahtera menggunakan Fleemo untuk memasarkan produknya. Mereka menginstal

aplikasi Fleemo sebagai *administrator* untuk memantau seluruh kendaraan/armada miliknya.

c. Tono sebagai yang bertanggung jawab atas pemantauan kendaraan membuka aplikasi, lalu membuat akun baru dengan membuka menu yang tersedia.
d. Tono mengisikan semua data yang diperlukan sebagai identitas perusahaan kemudian mendaftarkannya.
e. Masuk ke menu "*Home*" pada aplikasi.

2. *User* membuat akun sopir

   Skenario (gambar pada lampiran 11) :

   Tono memerintahkan 10 sopir utk 10 truk untuk menginstal aplikasi Fleemo khusus untuk sopir.

   a. Sopir tidak bisa membuat akunnya sendiri, melainkan harus dari aplikasi Fleemo untuk *administrator.*
   b. Dengan *database* sopir yang dimiliki perusahaan, Tono mendaftarkan semua sopir.
   c. Setelah semua didaftarkan, Tono membagikan *username* dan *password* tiap akun ke masing-masing sopir.

3. *User* mengubah data profil akunnya

   Skenario (gambar pada lampiran 12) :

   Perusahaan Alfasejahtera semakin berkembang dan dirasa perlu berpindah alamat ke lokasi yang lebih stategis.

   a. Setelah pindah alamat, Tono mengubah alamat yang tersimpan di aplikasi Fleemo ke alamat perusahaan saat ini.

b. Tono membuka menu "*Account*" lalu memilih menu *Company Account.*
c. Disana akan ditampilkan informasi profil perusahaan. Di bagian kanan atas, ada tombol *edit* profil, lalu Tono mengkliknya.
d. Di halaman *Edit* Profil, Tono menentukan kordinat lokasi perusahaan pada menu map.

4. *User* mengecek lokasi sopir
   Skenario (gambar pada lampiran 13) :
   Tono ingin mengetahui keberadaan sopir.
   a. Tono membutuhkan GPS, Fleemo memiliki GPS, dan Tono memutuskan untuk melihatnya menggunakan Fleemo.
   b. Tono dapat melihat keberadaaan para sopir baik yang sedang bekerja maupun yang tidak.

5. *User* membuat penugasan baru
   Skenario (gambar pada lampiran 14) :
   Toko Widya Utama melakukan pemesanan sejumlah barang pada Alfasejahtera.
   a. Setelah deal, Tono memberitahu Token yang harus diisikannya saat barang sudah sampai nanti.
   b. Tono masuk ke aplikasi dan membuka menu *task*, lalu membuat penugasan baru.
   c. Disana ada menu "check driver availability" untuk melihat lokasi sopir dan apakah sopir sedang bekerja.
   d. Setelah menentukan sopir untuk melakukan tugasnya, Toni mengirimkan penugasan kepada sopir untuk dikerjakan.

6. *User* menerima hasil penugasan sopir
   Skenario (gambar pada lampiran 15) :
   Saat armada menyelesaikan tugasnya, Muncul notifikasi dari aplikasi Fleemo.
   a. Tono membuka notifikasi tersebut, lalu masuk ke halaman untuk melihat penyelesaian tugas. Disana terlihat jumlah barang yang diharapkan sampai pada pelanggan dan berapa realisasinya.
   b. Tono melihat bahwa barang yang dibawa sopir sesuai dengan barang yang diterima pelanggan.
   c. Tono memberikan konfirmasi penyelesaian tugas.

7. *User* membatalkan penugasan baru
   Skenario (gambar pada lampiran 16) :
   Tono menerima pesanan dari UD. Maju Bersama.
   a. Tono membuat satu penugasan baru dan memilih sopir mana yang akan mengerjakan tugas tersebut.
   b. UD. Maju Bersama tiba-tiba membatalkan pesanan karena suatu hal dan sudah membuat kesepakatan dengan Alfasejahtera.
   c. Tono membuka menu "*Task*", lalu masuk ke sub menu "*On Going Task*"
   d. Tono mengklik tugas yang ingin dibatalkan lalu menekan tombol "*cancel delivery*".

4.5.2.2 *Smartphone* Sopir Armada

1. *User* mengganti informasi profil akun miliknya
   Skenario (gambar pada lampiran 17) :
   a. Judika tidak meregistrasi kartunya dengan baik, akibatnya kartu Judika diblokir. Kemudian Judika menggantinya dengan nomor baru.

b. Judika teringat dengan beberapa akun yang menggunakan nomor lamanya termasuk Fleemo dan ingin memperbaharuinya.
c. Judika melakukan *edit account* pada Fleemo dan mengedit bagian nomor teleponnya.
d. Judika mengklik *save* pada halaman fitur dan nomornya berhasil diperbaharui. Kini Judika dapat melanjutkan bisnisnya melalui Fleemo dengan nomor yang baru.

2. *User* mengecek penugasan yang baru diterima untuk dikerjakan
   Skenario (gambar pada lampiran 18) :
   a. Saat jam kerja, Judika membuka aplikasi *Fleemo Driver* untuk mengecek apakah ada penugasan baru untuknya.
   b. 20 menit kemudian, Judika melihat ada tugas baru.
   c. Judika melihat barang apa saja yang perlu diantar, kemudian di gudang mana barang harus diambil, dan kemana barang akan diantar.
   d. Karena tidak langsung mengerjakan penugasan tersebut, Judika menekan tombol *back* untuk kembali ke tampilan daftar penugasan yang belum dikerjakan.

3. Armada membawa lebih dari satu tugas sekaligus
   Skenario (gambar pada lampiran 19) :
   a. Judika memuat barang dalam *container* truknya dan memuat 3 penugasan.
   b. Judika membuka aplikasi Fleemo dan melihat daftar tugas yang ingin dikerjakan.

c. Judika memandai 3 tugas yang ingin dikerjakan lalu menekan tombol start.
d. Judika berangkat mengantarkan barang dimana pada peta terlihat 3 lokasi tujuan. Satu per satu tugas diantarkan dengan memilih tugas mana yang lebih dulu diantarkan.

4. *User* telah menyelesaikan pengantaran
   Skenario (gambar pada lampiran 20) :
   a. Judika telah sampai di tujuan lalu menurunkan barang di tempat pelanggan.
   b. Judika menyerahkan *smartphone*-nya agar diisikan oleh pelanggan.
   c. Pelanggan mengisikan jumlah barang yang diterima lalu mengisikan token yang diterima dari perusahaan.
   d. Pelanggan melakukan submit, lalu mengembalikan *smartphone* ke sopir.

5. *User* gagal meneyelesaikan tugas
   Skenario (gambar pada lampiran 21) :
   a. Pelanggan mengisikan jumlah barang yang diterima lalu mensubmit melalui *smartphone* sopir.
   b. Saat mengantarkan barang, judika mengantarkan barang kurang dari yang seharusnya, yakni telur yang pecah saat diperjalanan.
   c. Setelah diisikan, Judika kembali mengantarkan penugasan yang lain sembari menunggu konfirmasi dari perusahaan apakah ia yang akan mengantar kekurangan pesanan atau dialihkan ke sopir lain.

4.5.2.3 *Desktop* manajemen Perusahaan

1. Akses *website* Fleemo

   Skenario (gambar pada lampiran 22) :

   a. Perusahaan memiliki satu *monitor* besar di lobbi yang dimaksudkan agar disana ditampilkan peta persebaran lokasi armada.
   b. Tono menyambungkan *monitor* tersebut ke *PC* atau laptop yang bisa membuka *website* Fleemo.
   c. Tono membuka *login* dengan akunnya lalu memproyeksikannya ke *monitor*
   d. Disana terlihat *marker* yang mewakili tiap truk perusahaan.

4.6 *Prototype*

4.6.1 *Smartphone* Manajemen Perusahaan

Desain-desain untuk smartphone manajemen perusahaan dijabarkan pada Tabel 4-6. Prototype yang dirancang adalah fungsi-fungsi utama yang harus ada agar semua fungsi sesuai dengan user story pada Tabel 4-3 dapat dijalankan. Deain prototype merupakan hasil adopsi dari Ide 3 pada lampiran 8 dokumen ini.

Tabel 4.6 Prototype smartphone manajemen perusahaan

	
Log in	*Sign up*
	
Membuat penugasan baru	Melihat daftar penugasan yang belum diselesaikan

	
Melihat daftar penugasan yang sudah diselesaikan	Melihat peta persebaran armada perusahaan
	
Melihat persebaran armada dalam bentuk *list*	Melihat track yang dilalui saat pengantaran tugas

	
Melihat daftar armada yang dimiliki perusahaan	Melihat detail informasi akun perusahaan
	
Melihat *list* armada saat membuat penugasan baru	mengisi lokasi pelanggan dengan mentukan kordinatnya di *google map*

	
Melihat hasil pengiriman barang armada kemudian memutuskan langkah selanjutnya pada penugasan tersebut	Melihat detail pengiriman yang sedang diantarkan/ belum selesai

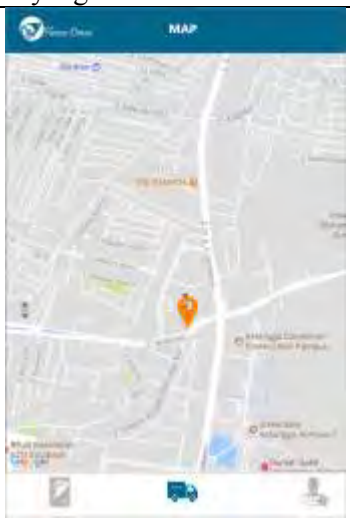
	
Melihat detail pengiriman yang sudah selesai	Melihat detail track lokasi pengantaran barang
	
Melihat detail armada yang sedang tidak bekerja	Mendaftarkan armada baru

	
Melihat detail informasi armada yang sudah didaftarkan	Mengedit informasi armada

4.6.2 *Smartphone* Sopir Armada

Desain-desain untuk smartphone sopir armada dijabarkan pada Tabel 4-7. Prototype yang dirancang adalah fungsi-fungsi utama yang harus ada agar semua fungsi sesuai dengan user story pada Tabel 4-4 dapat dijalankan. Deain prototype merupakan hasil adopsi dari Ide 3 pada lampiran 7 dokumen ini.

Tabel 4.7 Prototype smartphone sopir

	
Sign in	Melihat daftar penugasan yang belum diselesaikan
	
Melihat daftar penugasan yang sudah diselesaikan	Melihat lokasi terkini armada

	
Melihat detail akun armada	Melihat detail penugasan yang harus diselesaikan

	
Mengajukan penolakan penugasan	Rute perjalanan saat melakukan pengantaran barang
	
Memilih penugasan yang sudah sampai	Melihat daftar penugasan yang sudah selesai

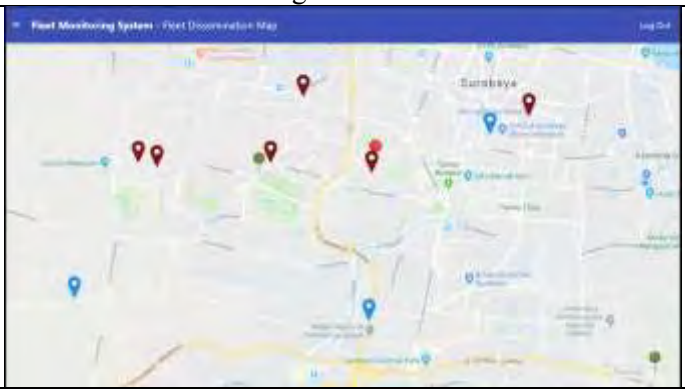
	
Melihat detail tugas yang sudah diselesaikan	Melihat detail rute dari tugas yang sudah diselesaikan
	
Pelanggan mengisikan detail barang yang sudah diterima	Pemberitahuan saat barang yang seharusnya dikirim dengan barang yang sampai jumlahnya sama

	
Saat pengiriman belum selesai dan perusahaan tetap menugaskan sopir yang sama untuk melakukan pengantaran	Saat pengiriman belum selesai dan perusahaan mengalihkan pengantaran ke sopir lain.

4.6.3 *Desktop* manajemen Perusahaan

Desain-desain untuk desktop manajemen perusahaan dijabarkan pada Tabel 4-8. Prototype yang dirancang adalah fungsi-fungsi utama yang harus ada agar semua fungsi sesuai dengan user story pada Tabel 4-5 dapat dijalankan. Deain prototype merupakan hasil adopsi dari Ide 3 pada lampiran 9 dokumen ini.

Tabel 4.8 Desktop manajemen perusahaan


Login website

Peta persebaran armada

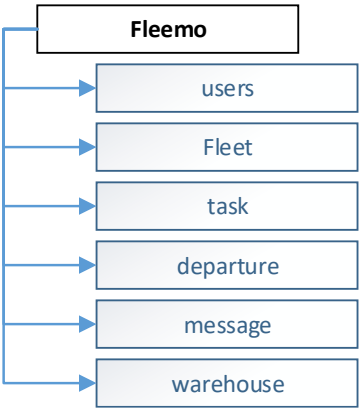
BAB 5
IMPLEMENTASI

Pada bab ini akan dijelaskan proses tahap implementasi aplikasi *Fleet Monitoring System*. Pembahasan yang dilakukan pada bab ini difokuskan pada lingkungan implementasi, struktur *database*, serta penulisan kode program.

5.1 Struktur *Database* Aplikasi

Aplikasi *Fleet Monitoring System* menggunakan *Realtime Database* yang disediakan oleh *Google Firebase*. Strukturnya tidak table seperti *SQL* namun berbentuk ranting pohon dan disimpan dalam objek *JSON*. Ketika menambahkan data ke pohon *JSON*, data tersebut akan menjadi node di struktur *JSON* yang ada dengan kunci. [11] Pada gambar dibawah ini, kotak yang berwarna hitam adalah pangkal dari pohon *JSON*, kotak berwarna jingga adalah *Parent Key*, kotak berwarna hijau adalah *Foreign Key*, dan kotak berwarna biru adalah *value* umum yang hanya dimiliki oleh masing-masing ranting.

Fleemo memiliki 6 ranting utama untuk menyimpan data-datanya. Ranting *user*s untuk menyimpan data perusahaan, ranting *Fleet* untuk menyimpan data sopir/armada, ranting *task* untuk menyimpan detail tugas, ranting *departure* untuk menyimpan data tugas-tugas yang dikerjakan dalam 1 keberangkatan, ranting *message* ditujukan untuk menyimpan pesan-pesan antara perusahaan dan armadanya, dan ranting *warehouse* untuk menyimpan data gudang. Struktur tersebut dapat dilihat pada gambar 5.1.



Gambar 5.1 Pohon JSON: Ranting utama

Ranting *user* menyimpan data *user*/perusahaan. 1 perusahaan masuk dalam 1 ranting *companyID*. Detail data 1 perusahaan adalah dalam cabang-cabang dibawah **ranting *companyID***. **Ranting *address*** untuk menyimpan alamat yang isinya adalah *latitude* dan *longitude*, **ranting *fullname*** untuk menyimpan nama perusahaan, **ranting *password*** untuk menyimpan *password* akun, **ranting *phone_number*** untuk menyipan nomor telepon perusahaan dan **ranting token** untuk meyimpan kode unik yang tercipta dalam setiap *login* di aplikasi. Apabila dalam companyID tidak memiliki 5 ranting tersebut, aplikasi akan error karena tidak bisa membaca data yang ia perlukan. Struktur ranting user dapat dilihat pada gambar 5.2.

Ranting *warehouse* menyimpan data gudang dan bercampur antara gudang 1 perusahaan dengan yang lainnya. Detail data tiap gudang adalah dalam cabang-cabang dibawah **ranting *warehouseID***. **Ranting *address*** untuk menyipan alamat gudang yang isinya adalah *latitude* dan *longitude*, **ranting *companyID*** berisi kode unik perusahaan sehingga yang bisa melihat detail 1 gudang adalah perusahaan dengan kode yang

sama dengan isi ranting *companyID*, dan **ranting *name*** untuk menyimpan nama gudang. Struktur ranting *warehouse* dapat dilihat pada gambar 5.3.

Ranting *Fleet* menyimpan data armada dan bercampur antara armada 1 perusahaan dengan yang lainnya. Detail data tiap armada adalah dalam cabang-cabang dibawah **ranting *FleetID***. **Ranting *address*** untuk menyipan alamat armada yang isinya adalah *latitude* dan *longitude*, **ranting *birthdate*** untuk menyimpan tanggal lahir sopir dengan format (mm/dd/yyyy)), **ranting *companyID*** berisi kode untuk perusahaan sehingga yang bisa melihat detail 1 armada adalah perusahaan dengan kode yang sama dengan isi ranting *companyID*, **ranting *departureID*** yang berisi kode pengantaran yang sedang dikerjakan. Ranting ini hanya memiliki nilai saat armada sedang bekerja, **ranting *email*** berisi *email* armada, **ranting *fullname*** berisi nama lengkap armada, **ranting *gender*** berisi jenis kelamin armada, **ranting isSignUp** berisi Boolean yang bernilai *false* ketika armada sudah didaftarkan oleh perusahaan namun belum pernah *login* sama sekali dan bernilai *true* apabila sudah pernah *login*. Hal ini perlu didefinisikan karena armada yang sudah didaftarkan perusahaan tidak memiliki otentikasi *Firebase* dan agar perusahaan tidak memberikan penugasan pada sopir yang belum pernah menggunakan akunnya. **Ranting *license*** berisikan nomor SIM sopir, **ranting *location*** berisikan lokasi terkini dari sopir yang direkam oleh aplikasi *Fleemo Driver*, ranting *password* berisikan *password* dari akun armada, **ranting phone** berisikan nomor telepon sopir, **ranting token** berisikan kode unik yang tercipta dalam setiap *login* di aplikasi. ***Truck plate*** dan ***truck name*** berisikan identitas truk yang digunakan sopir. Struktur ranting Fleet dapat dilihat pada gambar 5.4.

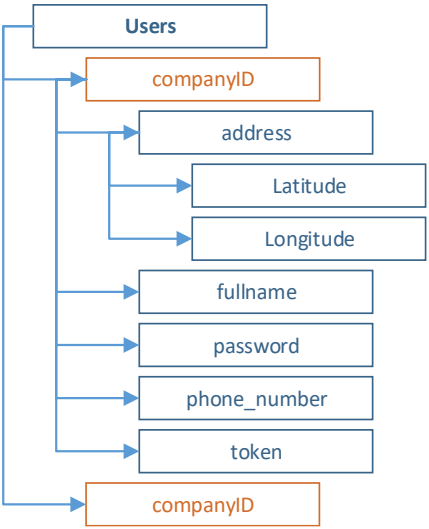
Ranting *Task* menyimpan data penugasan untuk banyak armada dan bercampur antara 1 perusahaan dengan yang lainnya. Detail penugasan ada dalam cabang-cabang dibawah **ranting *taskID***. **Ranting *address*** untuk menyimpan alamat pelanggan yang isinya adalah *latitude* dan *longitude*, **ranting *companyID*** berisi kode perusahaan sehingga yang bisa mengakses *task* ini adalah perusahaan dengan kode yang sama dengan isi ranting. **Ranting *customer*** berisikan nama pelanggan, **Ranting *departureID*** berisikan kode keberangkaatan untuk menandakan apakah tugas ini sedang dikerjakan atau belum sama sekali. **Ranting *FleetID*** berisi kode armada sehingga yang bisa mengakses dan mengerjakan *task* ini adalah armada degan kode yang sama dengan isi ranting. **Ranting *isCompleted*** berisikan nilai apakah tugas belum dikerjakan, dikerjakan dengan berhasil, dikerjakan namun jumlah barang yang sampai kurang, sisa penugasan dialihkan ke armada lain, atau sisa penugasan tetap dikerjakan oleh armada yang sama. **Ranting *location*** berisikan kordinat-kordinat lokasi yang dilewati armada saat mengantarkan pesanan. **Ranting *message*** berisikan barang-barang apa saja yang perlu diantarkan dalam 1 penugasan. 1 tugas berisikan **ranting *item*** yakni nama barang, **ranting target** yakni barang yang dipesan, dan **ranting actual** yakni barang yang sampai ke pelanggan. Ranting *actual* hanya terisi saat penugasan telah selesai dilakukan. **Ranting *warehouseLoc*** berisikan kordinat lokasi gudang, dan **ranting *warehouseName*** adalah nama gudang tempat pesanan pelanggan harus diambil. **Ranting *token*** adalah nilai unik yang dibuat oleh perusahaan dan hanya dibagikan kepada *customer* yang dituju. **Ranting *workType*** adalah jenis tugas yang dibedakan menjadi 2 yakni penugasan baru (*regular task*) dan penugasan yang dikerjakan karena

barang pada penugasan biasa tidak lengkap diantarkan (*rework task*). Struktur ranting Fleet dapat dilihat pada gambar 5.5.

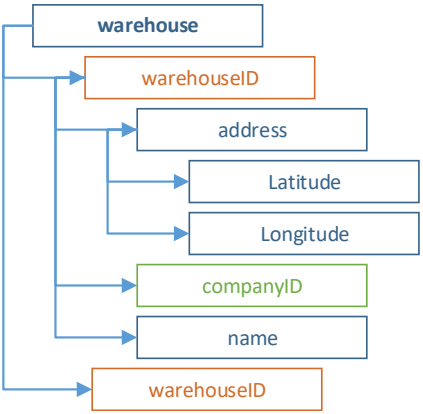
Ranting *message* menyimpan data pesan antara 1 perusahaan dengan perusahaan lainnya. Detail pesan ada dalam cabang-cabang dibawah **ranting *messageID***. Didalam ranting *message* adalah jajaran *companyID* untuk memisahkan pesan-pesan antar perusahaan. Kemudian didalam **ranting *companyID*** adalah jajaran **ranting *FleetID*** untuk mengelompokkan pesan-pesan. Desain ranting *message* sedikit berbeda karena pesan dapat sangat banyak dan jika digabungkan dapat membuat aplikasi bekerja terlalu berat. Dalam hal ini 1 aplikasi *FleemoCompany* hanya membaca yang ada dalam ranting *companyID* miliknya dan 1 aplikasi *FleemoDriver* hanya membaca yang ada dalam ranting *FleetID* miliknya. Saat ini *message* hanya berisikan pesan pembatalan penugasan, dan pengajuan pengalihan tugas. Di dalam ranting *messageID* ada **ranting *customer*** yang berisikan nama pelanggan tujuan, **ranting *departureID*** berisi kode pengantaran yang terisi apabila sopir ingin membatalkan pengantaran saat sedang dalam perjalanan. **Ranting *taskID*** berisi kode 1 penugasan yang terisi apabila sopir ingin membatalkan pengantaran saat tugas belum dikerjakan. **Ranting *isRead*** berisi apabila perusahaan sudah memberikan respon atas permintaan armada. Ranting ini bernilai *false* apabila armada belum membacanya dan bernilai *true* apabila sudah dibaca. **Ranting *location*** berisi lokasi armada saat mengajukan pengalihan tugas, **ranting *newDriver*** berisi sopir baru yang mengerjakan penugasan dan terisi apabila perusahaan menyetujui pengalihan tersebut. **Ranting *reason*** berisikan alasan mengapa sopir ingin mengalihkan tugasnya, **ranting *response*** berisikan *String* apakah perusahaan menyetujui, menolak atau memberitahu

pembatalan pengerjaan tugas. **Ranting *returnOfGood*** berisikan kapan armada akan mengembalikan barang yang seharusnya diantar. Ranting ini terisi apabila armada sudah dalam tahap pengantaran barang (memiliki *departureID*). Dan **ranting *type*** berisikan String apakah pengalihan penugasan ini adalah 1 *task* /1 *departure*. Struktur ranting *Fleet* dapat dilihat pada gambar 5.6.

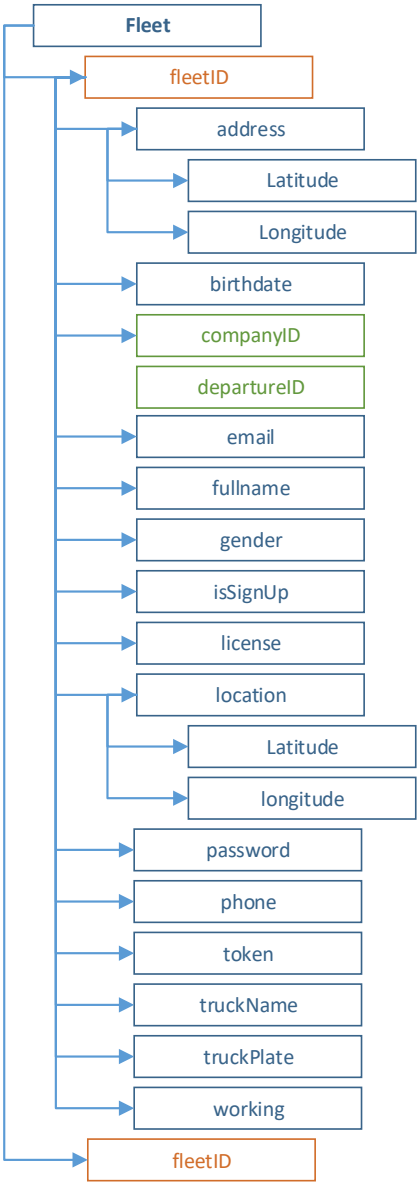
Ranting *departure* menyimpan banyak data keberangkatan antara 1 perusahaan dengan perusaan lainnya. Detail keberangkatan ada dalam cabang-cabang dibawah **ranting *departureID***. **Ranting *FleetID*** berisikan berisi kode armada sehingga yang bisa mengakses data keberangkatan ini adalah armada dengan kode yang sama dengan isi ranting. **Ranting *isDepartureCompleted*** berisikan nilai untuk menunjukkan apakah 1 keberangkatan sudah diselesaikan semua atau tidak, **ranting *isWorking*** berisikan nilai apakah 1 penugasan dalam keberangkatan ini sedang dikerjakan atau tidak. Apabila bernilai *true*, maka **ranting *taskWorking*** juga berisi *task* yang sedang diantarkan. **Ranting *task*** berisikan *task-task* apa saja yang diantarkan dalam 1 keberangkatan. **Ranting *isTaskCompleted*** berisikan nilai apakah 1 *task* tersebut sudah diantarkan atau belum. Apabila semua *isTaskCompleted* bernilai "*successfully completed*" atau "*unsuccessfully completed*", maka ranting *isDepartureCompleted* akan bernilai *true*. Struktur ranting Fleet dapat dilihat pada gambar 5.7.



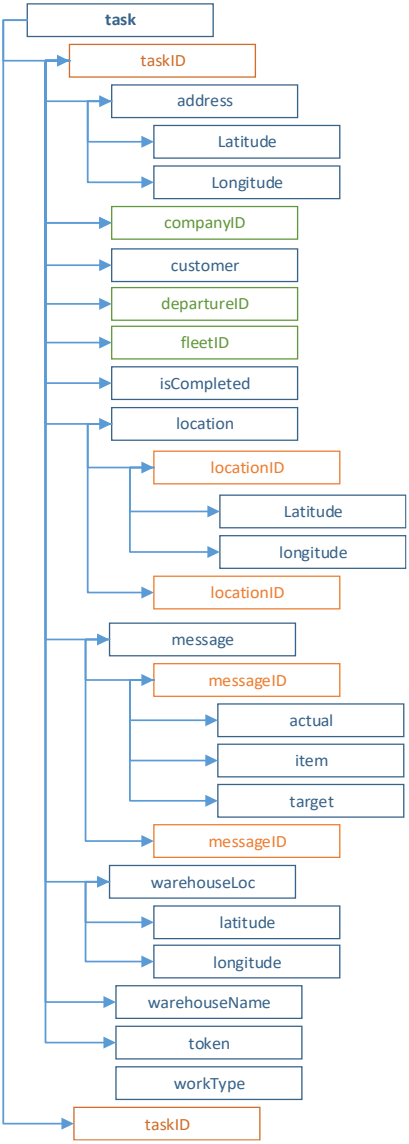
Gambar 5.2 Pohon JSON: Ranting users



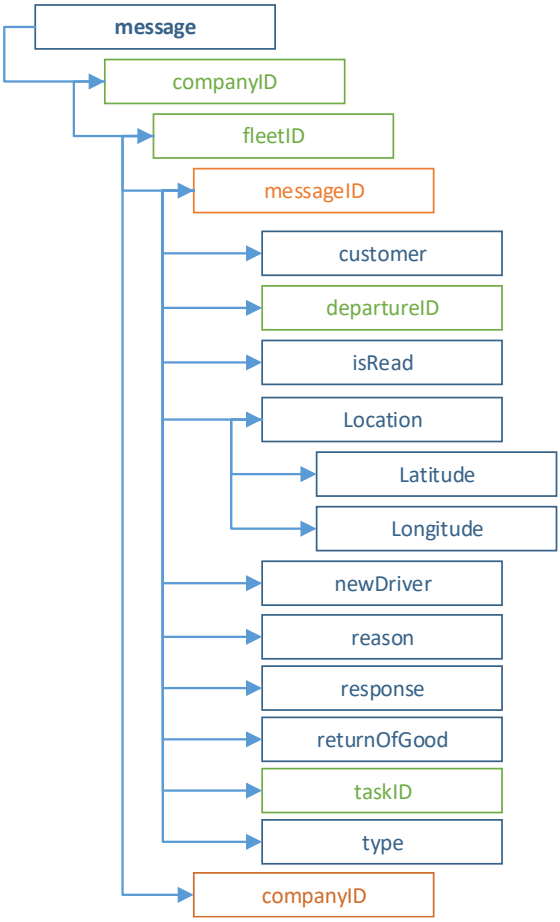
Gambar 5.3 Pohon JSON: Ranting warehouse



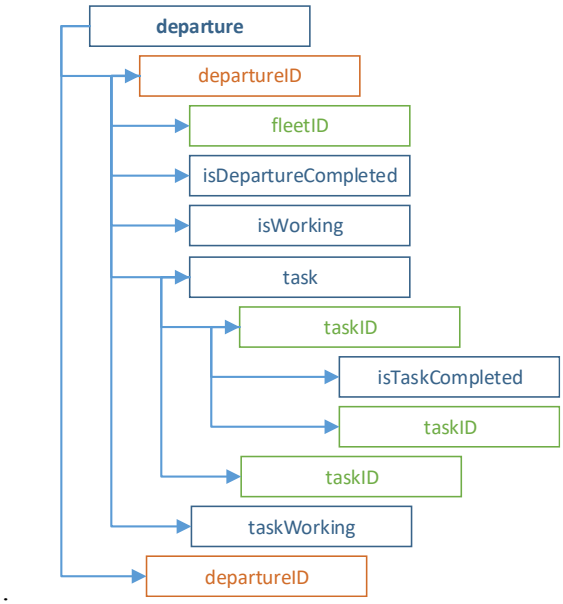
Gambar 5.4 Pohon JSON: Ranting fleet



Gambar 5.5 Pohon JSON: Ranting task



Gambar 5.6 Pohon JSON: Ranting message



Gambar 5.7 Pohon JSON: Ranting departure

Database pada Firebase tidak bisa diakses apabila tidak mengetahui API key dari database tersebut. Meskipun seseorang/ aplikasi memiliki API database, seseorang/aplikasi tersebut tidak dapat mengakses database secara sembarangan apabila tidak login/terotentikasi. Aturan untuk membatasi akses database hanya dapat dilakukan oleh orang yang terotentikasi didefinisikan pada **rule** firebase. Pada aplikasi sistem pemantau armada ini, 4 ranting yakni departure, task, warehouse, dan message dapat dibaca dan ditulis oleh aplikasi yang sudah terotentikasi. Sementara ranting fleet dan users dapat dibaca oleh aplikasi yang belum terotentikasi namun hanya bisa ditulis saat sudah terotentikasi. Hal ini dilakukan karena dibutuhkan

data email dari armada dan perusahaan saat login. *Rule* yang digunakan adalah seperti gambar 5.8 berikut:

```
{
  "rules": {

    "fleemo":{
      "departure": {
       ".read": "auth !== null",
       ".write": "auth !== null"
      },

      "message": {
       ".read": "auth !== null",
       ".write": "auth !== null"
      },

      "task": {
       ".read": "auth !== null",
       ".write": "auth !== null"
      },

      "temp_task": {
       ".read": "auth !== null",
       ".write": "auth !== null"
      },

      "fleet": {
       ".read": true,
       ".write": "auth !== null"
      },

      "users": {
        ".write": "auth !== null",
        ".read": true,
      },

      "warehouse": {
         ".read": "auth !== null",
         ".write": "auth !== null"
      }
    }
  }
}
```

Gambar 5.8 Rule firebase

5.2 *Smartphone* Manajemen Perusahaan

5.2.1 Lingkungan Aplikasi

Implementasi dari aplikasi *smartphone* manajemen perusahaan atau yang dinamakan ***Fleemo Company*** dilakukan menggunakan perangkat *virtual machine* dan *smartphone* nyata. Detail spesifikasi perangkat dapat dilihat pada tabel 5.1:

Tabel 5.1 Lingkungan aplikasi fleemo company

Virtual Machine	
Nama perangkat	Nexus 5
Processor	Multi-Core 4 CPUs
Memory	1536 MB
Sistem Operasi	Android API 24 x86 Nougat
Resolusi	5.2” 1080x1920 xxhdpi
Perangkat Nyata	
Nama perangkat	Samsung A5 2016
Processor	Samsung Exynos Octa 7580 1.60GHz
Memory	1827MB
Sistem Operasi	Android API 24 x86 Nougat
Resolusi	5.2” 1080x1920 xxhdpi

5.2.2 Fungsi-Fungsi Aplikasi

Pada bagian ini dijelaskan proses implementasi setiap fungsi aplikasi dengan potongan kode yang digunakan. IDE yang dipakai adalah **Android Studio 3.0.1.** Berikut adalah fungsi-fungsi yang dimiki oleh *Fleemo Company* sesuai pada user story pada tabel 4.3:

5.2.2.1 US-AM1-1: Mendaftarkan akun (*Sign up*)

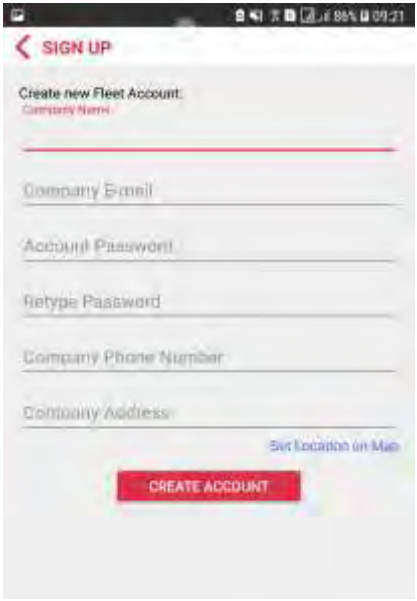
Saat aplikasi dubuka, *activity* yang pertama kali muncul adalah *FlashActivity* untuk mengecek apakah ada *user login* atau tidak.

Apabila tidak ada maka akan masuk ke *SignInActivity*, jika tidak maka akan langsung masuk ke halaman utama sambil mengirimkan *token* ke *Database*. Kode yang digunakan adalah pada gambar 5.9.

```
user = FirebaseAuth.getInstance().getCurrentUser();

        if(user != null){
            token =
FirebaseInstanceId.getInstance().getToken();
            uID = user.getUid();
            new
FireDataUserCompany().writeUserToken(uID, token, new
DatabaseReference.CompletionListener() {
                @Override
                public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                    startActivity(new
Intent(FlashActivity.this, HomeActivity.class));
                    finish();
                }
            });
        }
        else{
            startActivity(new
Intent(FlashActivity.this, SignInActivity.class));
            finish();
        }
    }
```

Gambar 5.9 Kode US-AM1-1: Mengecek user login

Jika ingin melakukan pendaftaran akun, maka kita masuk dari link yang disediakan di SignInAcitivity. *SignUpActicity* memiliki *layout* seperti gambar 5.10. Ada 5 data yang perlu untuk membuat akun *company*, yakni nama, *email*, kata sandi, nomor telepon, dan alamat. Terkhusus untuk alamat, data disimpan dalam kordinat *Latitude-Longitude*, namun ditampilkan dalam bentuk alamat yang dikonversi dengan *Geocoder*.



Gambar 5.10 US-AM1-1: Layout Sign-Up

Untuk mendaftarkan akun, aplikasi akan mengambil data-data pada EditText menggunakan kode pada gambar 5.11. Aplikasi menyimpan data alamat dalam format kordinat peta. Agar user dapat membacanya, kordinat dikonversikan menggunakan geocoder seperti pada gambar 5.12.

Sebelum akun dibuat, aplikasi mengecek kelengkapan informasi yang dimasukkan agar tidak ada *error* saat pengambilan data. Kode yang digunakan adalah pada gambar 5.13. Saat semua sudah terisi, barulah menggunakan fungsi dari *Firebase* untuk membuat akun menggunakan *email* dan *password*. Untuk membuatnya, langkah pertama adalah masuk ke halaman *Sign-in Method* dalam menu *Authentication* pada *Firebase Project*. Pastikan *provider Email*/*Password* dalam status ***enabled*** seperti pada gambar 5.14**.**

Langkah berikutnya adalah menggunakan kode berikut agar data *email* dan *password* didaftarkan ke *Firebase*. Saat pendaftaran berhasil, data akun dimasukkan ke *database* dengan ID yang sama dengan uID akun. Kode yang digunakan adalah pada gambar 5.15.

Firebase memiliki aturan dalam mendaftakan akun baru:

1. *Email* harus dituliskan dalam format penulisan *email*
2. *Email* harus unik
3. *Password* minimal 6 karakter

```
fullName = et_add_company_fullname.getText().toString();
email = et_add_company_email.getText().toString().trim();
password1 et_add_company_password1.getText().toString();
password2 = et_add_company_password2.getText().toString();
phoneNumber = company_phonenumber.getText().toString();
```

Gambar 5.11 US-AM1-1: Mengambil data EditText

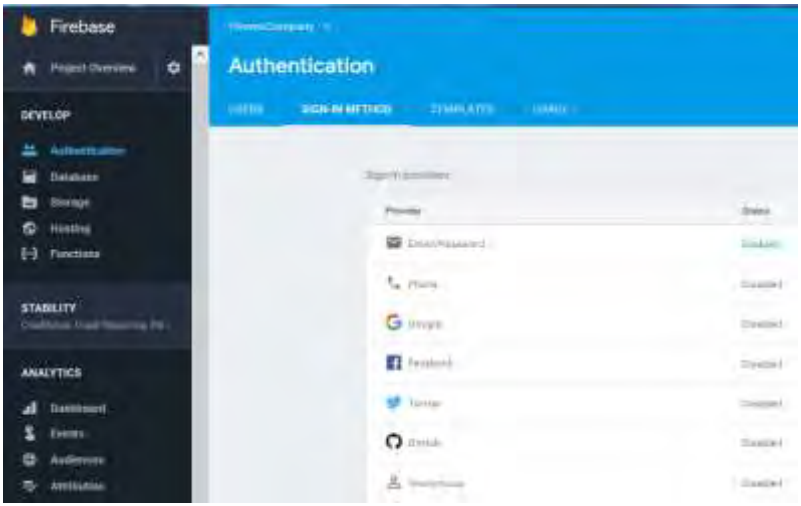
```
latitude =
Double.parseDouble(data.getStringExtra("latitude"));
longitude =
Double.parseDouble(data.getStringExtra("longitude"));
latLng = new LatLng(latitude, longitude);

Geocoder geocoder;
List<Address> addresses;
geocoder = new Geocoder(this, Locale.getDefault());
try {
    addresses = geocoder.getFromLocation(latitude,
longitude,1);
    address = addresses.get(0).getAddressLine(0);
    et_add_company_address.setText(address);
} catch (IOException e) {
    e.printStackTrace();
}
```

Gambar 5.12 US-AM1-1: Mengambil dan menkonversi data alamat

```
setValue();
if(TextUtils.isEmpty(fullName)){
    Toast.makeText(this,"Please enter Full Name",
Toast.LENGTH_SHORT).show();
    return;
}

if(TextUtils.isEmpty(email)){
    Toast.makeText(this,"insert Email",
Toast.LENGTH_SHORT).show();
    return;
}

else{
    String expression = "^[\\w\\.-]+@([\\w\\-]+\\.)+[A-
Z]{2,4}$";
    Pattern pattern = Pattern.compile(expression,
Pattern.CASE_INSENSITIVE);
    Matcher matcher = pattern.matcher(email);

    if(!matcher.matches()){
        Toast.makeText(this,"email format invalid",
Toast.LENGTH_SHORT).show();
        return;
    }
}

if(TextUtils.isEmpty(password1)){
    Toast.makeText(this,"insert Password",
Toast.LENGTH_SHORT).show();
    return;
}

else if(password1.length()<6){
    Toast.makeText(this,"Password minimum character:
6", Toast.LENGTH_SHORT).show();
}

if(TextUtils.isEmpty(password2)){
    Toast.makeText(this,"re-insert your password",
Toast.LENGTH_SHORT).show();
    return;
}

if(TextUtils.isEmpty(phoneNumber)){
    Toast.makeText(this,"insert Phone Number",
Toast.LENGTH_SHORT).show();
    return;
}

if(latLng==null){
    Toast.makeText(this,"Set address with a map",
```

```
Toast.LENGTH_SHORT).show();
    return;
}

if(!password1.equals(password2)){
    Toast.makeText(this,"the passwords you typed didn't
match" , Toast.LENGTH_SHORT).show();
    return;
}

else{
    //Sign Up
}
```

Gambar 5.13 US-AM1-1: Cek EditText yang kosong



Gambar 5.14 US-AM1-1: Firebase sign method

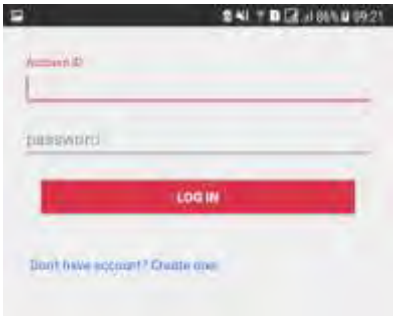
```
try{
mAuth.createUserWithEmailAndPassword(email,password1).a
ddOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull
Task<AuthResult> task) {
            if(task.isSuccessful()){
                Toast.makeText(SignUpActivity.this,
"Account Registered successfully",
Toast.LENGTH_SHORT).show();
                final FirebaseUser user =
mAuth.getCurrentUser();

                new
FireDataUserCompany().writeUser(user.getUid(),
fullName, email, password1, phoneNumber, latLng, new
DatabaseReference.CompletionListener() {
                    @Override
                    public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                        if(databaseError == null){
                            startActivity(new
Intent(SignUpActivity.this, HomeActivity.class));
                            finish();
                        }
                        else{

Toast.makeText(SignUpActivity.this, "Registration
error, please try again : " +
databaseError.getDetails(), Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            }
            else {
                Toast.makeText(SignUpActivity.this,
"Registration error, please try again : " +
task.getException(), Toast.LENGTH_SHORT).show();
                return;
            }
        }
    });
}
catch (Throwable t){
    t.printStackTrace();
    AlertDialog alertDialog = new
AlertDialog.Builder(this).create();
```

```
    alertDialog.setTitle("Error");
    alertDialog.setMessage("Application running on
Android 4.4 and above");
    alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL,
"OK",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface
dialog, int which) {
                    dialog.dismiss();
                }
            });
    alertDialog.show();
    return;
}
```

Gambar 5.15 US-AM1-1: Fungsi Sign up

5.2.2.2 US-AM1-2: *Sign in*

SignInActivity merupakan *activity* yang memungkinkan aplikasi untuk *sign in* dan mengakses seluruh data yang diizinkan pada *firebase*. Tampilan *sign in* adalah seperti gambar 5.16.



Gambar 5.16 US-AM1-2: Tampilan Sign in

Pada aplikasi Fleemo *Company*, hanya *email* yang terdaftar sebagai *company* atau pada *database* berada pada ranting ***user*** saja yang bisa masuk sedangkan *email* yang berada pada ranting ***fleet*** tidak boleh. Untuk itu pada awal masuk *activity* ini, aplikasi mengakses data *firebase* untuk mengambil *email* apa saja yang terdaftar pada ranting *user*. *Email* tersebut

dimasukkan ke dalam ***List emailList*** menggunakan kode pada gambar 5.17.

```
private void getEmail(){
    emailEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                String tempEmail =
postSnapshot.child(FireDataUserCompany.EMAIL).getValue(
).toString();
                emailList.add(tempEmail);
            }
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataUserCompany().ref.addValueEventListener(emailEv
entListener);
}
```

Gambar 5.17 US-AS1-2: Mengambil email company

Kemudian saat tombol *login* ditekan, maka teks dari *EditText* diambil, lalu dicek apabila kosong maka tidak melakukan fungsi apapun, apabila berisi, selanjutnya adalah menjalankan fungsi signIn yang disediakan oleh *Firebase* seperti kode pad gambar 5.18. Aplikasi akan mengecek apakah *email* yang diketik *user* ada di dalam *emailList*. Apabila tidak ada, maka muncul *Toast* yang mengatakan "*Email not registered*", apabila ada, maka fungsi yang dijalankan adalah *signInWithEmailAndPassword* karena *signIn provider* pada *firebase* adalah *Email*/*Password*. Jika *sign in* tetap gagal, kemungkinan kesalahan adalah *password* yang salah sehingga muncul *Toast* yang mengatakan "*Password is incorrect*", dan apabila berhasil maka masuk ke class *HomeActivity*.

```
private void setValue(){
    email = et_id.getText().toString();
    password = et_password.getText().toString();
}

@OnClick(R.id.btn_login) public void login(){
    setValue();
    if(TextUtils.isEmpty(email)){
        Toast.makeText(this, "Please enter your ID",
Toast.LENGTH_SHORT).show();
        return;
    }
    else if(TextUtils.isEmpty(password)){
        Toast.makeText(this, "Please enter Password",
Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        for(int i=0;i<emailList.size();i++){
            if(email.equals(emailList.get(i))){
                Log.d("tsss","emailList: " +
emailList.get(i) + "&& email: " + email + " : " +
password);
                mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull
Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            token =
FirebaseInstanceId.getInstance().getToken();
                            uID =
mAuth.getCurrentUser().getUid();
                            email =
mAuth.getCurrentUser().getEmail();
                            new
FireDataUserCompany().writeUserToken(uID, token, new
DatabaseReference.CompletionListener() {
                                @Override
                                public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                                    new
FireDataUserCompany().writeUserPassword(uID, password,
new DatabaseReference.CompletionListener() {
                                        @Override
                                        public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {

startActivity(new Intent(SignInActivity.this,
```

```
HomeActivity.class));
                                                    finish();
                                                }
                                            });
                                        }
                                    });

                                }
                                else {

Toast.makeText(SignInActivity.this, "Password is
incorrect", Toast.LENGTH_LONG).show();
                                    return;
                                }
                            }
                        }).addOnFailureListener(new
OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull
Exception e) {

Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_SHORT).show();
                            }
                        });
                        break;
                    }
                    if(i==emailList.size()-1){
                        if(!email.equals(emailList.get(i))){

if(FirebaseAuth.getInstance().getCurrentUser()==null){
                                Toast.makeText(this, "Email not
registered", Toast.LENGTH_SHORT).show();
                                return;
                            }
                        }
                    }
                }
            }
        }
```

Gambar 5.18 US-AMI1-2: Sign in dengan email dan password

5.2.2.3 US-AM1-3: Mendaftarkan akun armada

Untuk mendaftarkan armada, *user* akan harus masuk ke *activity* ***Fleet*Registration**. Data awal yang ditanyakan adalah nama lengkap sopir, *email* sopir, *password* default, nomor lisesnsi

SIM sopir, nomor telepon sopir, jenis kelamin sopir, tanggal lahir sopir, alamat tinggal sopir yang harus ditentukan dengan kordinat map, nama truk, dan nomor plat truk. Untuk menentukan kordinat alamat sopir, maka masuk ke *activity* **Account_Set_Map** pada gambar 5.19.

Gambar 5.19 US-AMI1-3: Menentukan lokasi di Map

Ujung dari pin diletakkan tepat di tengah layar untuk mempermudah *user* menentukan lokasinya. Saat posisi sudah tepat, *user* mengklik ***set location*** untuk mengambil data kordinat dan kembali ke *activity Fleet*Registration. Kode yang digunakan adalah seperti gambar 5.20.

```
OnClick(R.id.ivSetLocation) public void setLocation(){
    mMap.clear();

    VisibleRegion visibleRegion =
mMap.getProjection().getVisibleRegion();
    Point x =
mMap.getProjection().toScreenLocation(visibleRegion.far
Right);
    Point y =
mMap.getProjection().toScreenLocation(visibleRegion.nea
rLeft);
    Point centerPoint = new Point(x.x / 2, y.y / 2);
    centerFromPoint =
mMap.getProjection().fromScreenLocation(centerPoint);
    //mMap.addMarker(new
MarkerOptions().position(centerFromPoint)).setTitle("ok
e");
    Log.d("tsss","center from point: " +
centerFromPoint.toString());


    Geocoder geocoder;
    List<Address> addresses;
    geocoder = new Geocoder(this, Locale.getDefault());
    try {
        addresses =
geocoder.getFromLocation(centerFromPoint.latitude,
centerFromPoint.longitude,1);
        String address =
addresses.get(0).getAddressLine(0);
        Log.d("tsss", "Cordinate: "+
centerFromPoint.latitude + " : " +
centerFromPoint.longitude);
        Log.d("tsss", "address: " + address);

        Intent intent = new Intent();
        intent.putExtra("latitude",
String.valueOf(centerFromPoint.latitude));
        intent.putExtra("longitude",
String.valueOf(centerFromPoint.longitude));
        setResult(Activity.RESULT_OK, intent);
        finish();
    }

    catch (IOException e) {
        e.printStackTrace();
    }
}
```

Gambar 5.20 US-AM1-3: Set lokasi sopir

Setelah semua data diisikan, *user* mengklik tombol "Create *Fleet Account*". Tampilan dari *activity* ini seperti pada gambar 5.21.

Gambar 5.21 US-AM1-3: Tampilan aktivitas FleetRegistration

Setelah diklik, aplikasi mengambil semua teks dari *EditText* kemudian memastikan semua data telah terisi agar tidak terjadi error. Selain mengecek apakah data sudah terisi, pada kolom tertentu seperti jenis kelamin dicek apakah kata-kata yang diisikan sudah sesuai ("*male*"/"*female*"). Pada kolom tanggal lahir dicek apakah sudah sesuai dengan format tanggal (mm/dd/yyyy). Dan apakah karakter password sudah minimal 6 karakter. Kode yang digunakan adalah pada gambar 5.22.

```
private void setValue(){
    fullname = et_fullname.getText().toString();
    email = et_email.getText().toString();
    password1 = et_password1.getText().toString();
    password2 = et_password2.getText().toString();
    license = et_License.getText().toString();
    phoneNumber = et_phonenumber.getText().toString();
    gender = et_gender.getText().toString();
    birthDate = et_birthdate.getText().toString();
    truckName = et_truck_name.getText().toString();
    truckPlate = et_truck_plate.getText().toString();
}

private boolean validate(String date){
    String expression = "(0?[1-9]|1[012])+[/.-]+(0?[1-
9]|[12][0-9]|3[01])+[/.-]+((19|20)\\d\\d)";
    Matcher matcher =
Pattern.compile(expression).matcher(date);

    if(matcher.matches()){
        String value = matcher.group(0);
        int month = Integer.parseInt(value.substring(0,2));
        int day = Integer.parseInt(value.substring(3,5));
        int year = Integer.parseInt(value.substring(6));
        Log.d("tsss","Value: "+ month + " : " + day + " : "
+ year);

        if ((day<0 || day>30) && (month==4 || month==6 ||
month==9 ||month==11)) {

            Log.d("tsss", "birthdate check: false month
4,6,9,11");
            return false;
        }
        else if((day<0 || day>31) && (month==1 || month==3
|| month==5 ||month==7 ||month==8 ||month==10
||month==12)){
            Log.d("tsss", "birthdate check: false month
1,3,5,7,8,10,12");
            return false;
        }
        else if (month==2) {
            if(year % 4==0){
                if(day<0 || day>29){
                    Log.d("tsss", "birthdate check: false
month 2 date 30,31");
                    return false;
                }
                else{
                    return true;
                }
            }
```

```
            else{
                if(day>28){
                    Log.d("tsss", "birthdate check: false
month 2 date 29,30,31");
                    return false;
                }
                else{
                    return true;
                }
            }
        }
        else if(month<0 || month >12){
            return false;
        }
        else{
            return true;
        }
    }
    else{
        Log.d("tsss", "matcher no match");
        return false;
    }
}

@OnClick(R.id.btn_add_fleet) public void createFleet(){
    setValue();

    if(TextUtils.isEmpty(fullname)){
        Toast.makeText(this, "Please enter Full Name",
Toast.LENGTH_SHORT).show();
        return;
    }

    if(TextUtils.isEmpty(email)){
        Toast.makeText(this, "Please enter Email",
Toast.LENGTH_SHORT).show();
        return;
    }

    else{
        String expression = "^[\\w\\.-]+@([\\w\\-]+\\.)+[A-
Z]{2,4}$";
        Pattern pattern = Pattern.compile(expression,
Pattern.CASE_INSENSITIVE);
        Matcher matcher = pattern.matcher(email);

        if(!matcher.matches()){
            Toast.makeText(this, "email format invalid",
Toast.LENGTH_SHORT).show();
            return;
        }
    }
```

```
    if(TextUtils.isEmpty(password1)){
        Toast.makeText(this, "Please enter Password",
Toast.LENGTH_SHORT).show();
        return;
    }

    else if(password1.length()<6){
        Toast.makeText(this, "Password minimum character:
6", Toast.LENGTH_SHORT).show();
    }

    if(TextUtils.isEmpty(password2)){
        Toast.makeText(this, "Please re-enter Password",
Toast.LENGTH_SHORT).show();
        return;
    }

    if(TextUtils.isEmpty(license)){
        Toast.makeText(this, "Please enter License",
Toast.LENGTH_SHORT).show();
        return;
    }

    if(TextUtils.isEmpty(phoneNumber)){
        Toast.makeText(this, "Please enter Phone Number",
Toast.LENGTH_SHORT).show();
        return;
    }

    if(latLng==null){
        Toast.makeText(this, "Set address with a map",
Toast.LENGTH_SHORT).show();
        return;
    }

    if(TextUtils.isEmpty(address)){
        Toast.makeText(this, "Please enter Address",
Toast.LENGTH_SHORT).show();
        return;
    }

    if(TextUtils.isEmpty(gender)){
        Toast.makeText(this, "Please enter Gender",
Toast.LENGTH_SHORT).show();
        return;
    }

    else{
        String _gender = gender.trim().toLowerCase();
        if(!_gender.equals("male") &&
!_gender.equals("female")) {
            Toast.makeText(this, "gender format invalid",
```

```
Toast.LENGTH_SHORT).show();
            return;
        }
    }

    if(TextUtils.isEmpty(birthDate)){
        Toast.makeText(this, "Please enter Birth Date",
Toast.LENGTH_SHORT).show();
        return;
    }

    else{
        if(!validate(birthDate)){
            Toast.makeText(this, "birth date format
invalid", Toast.LENGTH_SHORT).show();
            return;
        }
    }

    if(TextUtils.isEmpty(truckName)){
        Toast.makeText(this, "Please enter Truck Name",
Toast.LENGTH_SHORT).show();
        return;
    }

    if(TextUtils.isEmpty(truckPlate)){
        Toast.makeText(this, "Please enter Truck Plate",
Toast.LENGTH_SHORT).show();
        return;
    }

    if(!password1.equals(password2)){
        Toast.makeText(this, "The password you typed didn't
match", Toast.LENGTH_SHORT).show();
        return;
    }

    else{
        if(emailList.contains(email)){
            Toast.makeText(this, "this email is already
registered", Toast.LENGTH_SHORT);
        }
        else{
            addFleetFirebase();
        }
    }
}
```

Gambar 5.22 US-AM1-3: Memastikan semua data telah terisi

Apabila semua diisikan dan alamat ditentukan dari map (tidak diketik manual), barulah masuk ke fungsi untuk membuat *database* armada dengan kode pada gambar 5.23.

```
private void addFleetFirebase(){
    new FireDataUserFleet().writeUser(adminuID,
fullname, email, password1, license, phoneNumber,
latLng,
            gender, birthDate, truckName, truckPlate,
new DatabaseReference.CompletionListener() {
                @Override
                public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                    if(databaseError == null){

Toast.makeText(FleetRegistration.this, "Registration
Completed ", Toast.LENGTH_SHORT).show();
                        finish();
                    }
                    else{

Toast.makeText(FleetRegistration.this, "Registration
error : "
                                    +
databaseError.getDetails(), Toast.LENGTH_SHORT).show();
                    }
                }
            });
}
```

Gambar 5.23 US-AM1-3: Menambahkan data armada ke Firebase

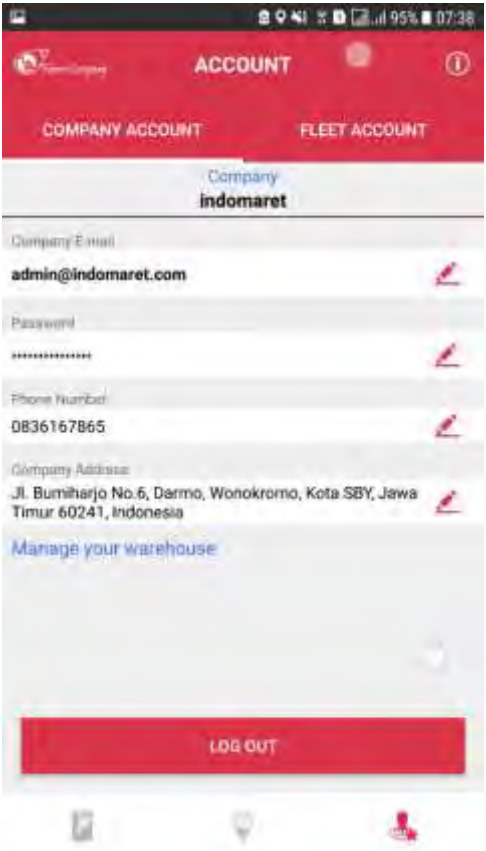
5.2.2.4 US-AMI-4: Mengedit informasi akun perusahaan

Informasi akun perusahaan ditampilkan dalam *activity CompanyAccount* dengan data yang diambil dari *firebase* dengan kode pada gambar 5.24. Data tersebut diambil pada ranting *users*→*usersID*, kemudian mengambil semua data yang ada didalamnya

```
private void getCompanyData(){
    singleEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            tmpKey = user.getUid();
            Log.d("tsss" , "current key:" + tmpKey);

            ModelCompany company =
dataSnapshot.getValue(ModelCompany.class);
            ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.cla
ss);

            Geocoder geocoder;
            List<Address> _address;
            geocoder = new Geocoder(getContext(),
Locale.getDefault());
            try {
                _address =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                address =
_address.get(0).getAddressLine(0);
            } catch (IOException e) {
                e.printStackTrace();
            }

            tv_ID_Fill.setText(company.getFullname());
            txt_email.setText(company.getEmail());
            txt_address.setText(address);
            txt_password.setText("companyPassword");

txt_phone.setText(company.getPhonenumber());
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataUserCompany().ref.child(user.getUid()).addListe
nerForSingleValueEvent(singleEventListener);
}
```

Gambar 5.24 US-AM1-4: Mengambil data perusahaan

Setelah berhasil diambil dari database, data-data tersebut ditampilkan dalam *user interface* dari *AccountActivity* seperti pada gambar 5.25.



Gambar 5.25 US-AM1-4: Tampilan company account

Untuk mengedit masing-masing data, maka *user* harus menekan ikon pensil, kecuali nama perusahaan yang harus menekan text ***Company***, lalu masuk ke *activity* yang memungkin mengedit

masing-masing data. Berikut adalah potongan kode untuk mengedit masing-masing data:

1. *Email* perusahaan
   Edit *email* tidak cukup hanya memperbaharui yang ada di *database*, namun ditambah memperbaharuinya pada otentikasi. Untuk memperbahaui ini, edit harus pada sesi yang sama dengan *login*. Jadi kalau edit *email* dilakukan saat aplikasi sudah pernah ditutup tetapi belum log out, edit *email* tidak bisa berfungsi. Itu adalah kebijakan dari *Firebase* sendiri. Kode pada gambar 5.26 digunakan untuk mengedit *email* perusahaan. Fungsi *updateDatabaseEmail* digunakan untuk mengedit *email* pada *database* dan fungsi *updateAuthEmail* untuk mengedit *email* pada otentikasi.
   Email yang dimasukkan user juga akan dicek apakah sudah sesuai dengan format penulisan email.

2. *Password* akun
   Password akun bisa diganti apabila sudah mengisikan *password* lama pada *EditText* dan cocok dengan *password* yang ada di *database*. Selain itu, *password* baru harus diketikkan 2 kali untuk memastikan *user* mengetikkan *password* sesuai dengan yang ia inginkan. Kode yang diguanakan pada gambar 5.27. Sama dengan mengedit *email* perusahaan, *password* akun harus diedit pada *database* dan otentikasi, juga harus diedit pada sesi yang sama dengan *login*. Kode yang digunakan adalah pada gambar 5.28. Email yang dimasukkan user akan dicek langsung dari Firebase dengan batasan password adlah minimal 6 karakter

```
private void updateDatabaseEmail(){
    new
FireDataUserCompany().writeUserEmail(user.getUid(),
newEmail, new
DatabaseReference.CompletionListener() {
        @Override
        public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
            if(databaseError == null){
                finish();
            }
            else{

Toast.makeText(Company_Edit_Email.this, "Error
while updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
            }
        }
    });
}

private void updateAuthEmail(){

user.updateEmail(newEmail).addOnCompleteListener(ne
w OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void>
task) {
            if(task.isSuccessful()){
                updateDatabaseEmail();
                finish();
            }
            else{

Toast.makeText(getApplicationContext(),"Error
updating. Please log out then log in again",
Toast.LENGTH_LONG).show();
            }
        }
    }).addOnFailureListener(new OnFailureListener()
{
        @Override
        public void onFailure(@NonNull Exception e)
{
            return;
        }
    });
}

@OnClick(R.id.btn_save_company_email) public void
saveEmail(){
```

```
    newEmail = et_edit_email.getText().toString();

    if(TextUtils.isEmpty(newEmail)){
        Toast.makeText(this, "new email is empty",
Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        String expression = "^[\\w\\.-]+@([\\w\\-
]+\\.)+[A-Z]{2,4}$";
        Pattern pattern =
Pattern.compile(expression,
Pattern.CASE_INSENSITIVE);
        Matcher matcher =
pattern.matcher(newEmail);

        if(matcher.matches()){
            updateAuthEmail();
        }
        else{
            Toast.makeText(this, "input format
invalid", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Gambar 5.26 US-AM1-4: Edit email perusahaan

3. Nomor telepon perusahaan

   Untuk mengedit nomor telepon perusahaan, langkahnya hanya mengambil teks yang ada dalam *EditText* lalu menyimpannya dalam *database*. Tidak ada aturan khusus yang membatasi user untuk mengisi nomor telepon sesuai aturan yang ada., Kode yang digunakan adalah pada gambar 5.29.

```
private void getOldPassword(){
    valueEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            ModelCompany company =
dataSnapshot.getValue(ModelCompany.class);
            savedOldPassword = company.getPassword();
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataUserCompany().ref.child(user.getUid()).addListene
rForSingleValueEvent(valueEventListener);
}

@OnClick(R.id.btn_save_company_password) public void
savePassword(){
    oldPassword = et_old_password.getText().toString();
    password1 = et_new_password1.getText().toString();
    password2 = et_new_password2.getText().toString();

    if(TextUtils.isEmpty(oldPassword)){
        Toast.makeText(this, "old password is empty",
Toast.LENGTH_SHORT).show();
        return;
    }
    else if(!oldPassword.equals(savedOldPassword)){
        Toast.makeText(this, "old password didn't match",
Toast.LENGTH_SHORT).show();
    }
    else if(TextUtils.isEmpty(password1)){
        Toast.makeText(this, "Please enter new password",
Toast.LENGTH_SHORT).show();
        return;
    }
    else if(password1.length()<6){
        Toast.makeText(this, "Password minimum character:
6", Toast.LENGTH_SHORT).show();
    }
    else if(TextUtils.isEmpty(password2)){
        Toast.makeText(this, "retype your new password",
Toast.LENGTH_SHORT).show();
        return;
    }
    else if(!password1.equals(password2)){
        Toast.makeText(this, "new password didn't match",
```

```
Toast.LENGTH_SHORT).show();
            return;
        }
        else{
            updateAuthPassword();
        }
}
```

Gambar 5.27 US-AM1-4: Cek input password

4. alamat perusahaan
   Untuk mengedit alamat perusahaan, langkahnya adalah mengambil kordinat lokasi menggunakan Aktivitas *Account*_Set_Map, lalu menyimpannya dalam *database*, menggunakan kode pada gambar 5.30.

5. nama perusahaan
   Untuk mengedit nama perusahaan, langkahnya hanya mengambil teks yang ada dalam *EditText* lalu menyimpannya dalam *database*, menggunakan kode pada gambar 5.31.

```
private void updateAuthPassword(){

user.updatePassword(password1).addOnCompleteListene
r(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void>
task) {
            if(task.isSuccessful()){
                Log.d("tsss","authentication update
success");
                updateDatabasePassword();
            }
            else{

Toast.makeText(getApplicationContext(),"Error
updating. Please log out then log in again",
Toast.LENGTH_LONG).show();
                return;
            }
        }
    });
}

private void updateDatabasePassword(){
    new
FireDataUserCompany().writeUserPassword(user.getUid
(), password1, new
DatabaseReference.CompletionListener() {
        @Override
        public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
            if(databaseError == null){
                Log.d("tsss", "database update
success");
                finish();
            }
            else{

Toast.makeText(Company_Edit_Password.this, "Error
while updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

Gambar 5.28 US-AM1-4: Edit password perusahaan

```
@OnClick(R.id.btn_save_company_phonenumber) public
void savePhone(){
    newPhone =
et_edit_phonenumber.getText().toString();

    if(TextUtils.isEmpty(newPhone)){
        Toast.makeText(this, "new phone number is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserCompany().writeUserPhoneNumber(user.get
Uid(), newPhone, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Company_Edit_Phone.this, "Error
while updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
    finish();
}
```

Gambar 5.29 US-AM1-4: Edit nomor telepon perusahaan

```
@Override protected void onActivityResult(int
requestCode, int resultCode, Intent data){
    if(requestCode==1){
        if(resultCode == Activity.RESULT_OK){
            latitude =
Double.parseDouble(data.getStringExtra("latitude"))
;
            longitude =
Double.parseDouble(data.getStringExtra("longitude")
);
            latLng = new LatLng(latitude,
longitude);

            Geocoder geocoder;
            List<Address> addresses;
            geocoder = new Geocoder(this,
Locale.getDefault());
            try {
                addresses =
geocoder.getFromLocation(latitude, longitude,1);
                address =
addresses.get(0).getAddressLine(0);
                et_edit_address.setText(address);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

@OnClick(R.id.tv_setMap) public void setMap(){
    Intent intent = new
Intent(Company_Edit_Address.this,
Account_Set_Map.class);
    startActivityForResult(intent, 1);
}

@OnClick(R.id.btn_save_company_address) public void
saveAddress(){
    if(latLng==null){
        Toast.makeText(this, "Set address with a
map", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserCompany().writeUserAddress(user.getUid(
), latLng, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
```

```
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Company_Edit_Address.this, "Error
while updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

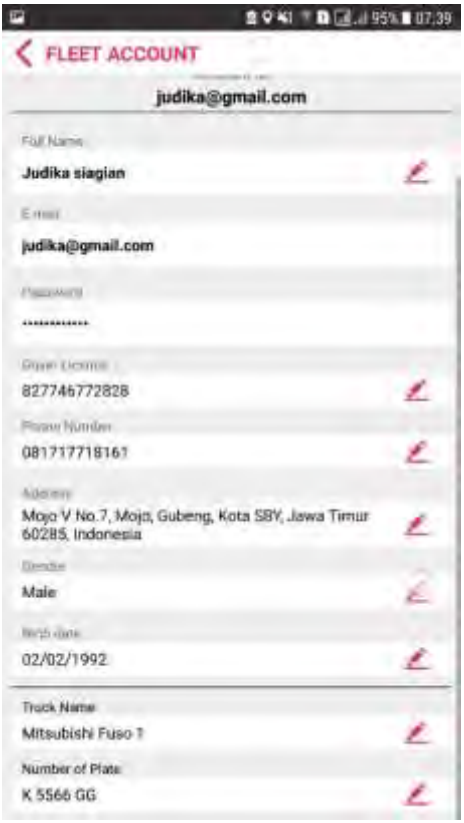
Gambar 5.30 US-AM1-4: Edit alamat perusahaan

```
@OnClick(R.id.btn_save_company_fullname) public
void saveName(){
    newName =
et_edit_fullname.getText().toString();

    if(TextUtils.isEmpty(newName)){
        Toast.makeText(this, "new company name is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserCompany().writeUserFullName(user.getUid
(), newName, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Company_Edit_Name.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
    finish();
}
```

Gambar 5.31 US-AM1-4: Edit nama perusahaan

5.2.2.5 US-AM1-5: Mengedit informasi akun armada

Detail informasi akun ditampilkan dalam *activity* *FleetAccountDetail* dengan tampilan pada gambar 5.33. Data terebut diambil dari *firebase* menggunakan kode pada gambar 5.33. Data-data yang ditampilkan adalah nama lengkap armada, *email*, nomor lisensi SIM, nomor telepon, alamat lengkap, jenis kelamin, tanggal lahir, nama truk, dan nomor plat truk.



Gambar 5.32 US-AM1-5: Tampilan fleet account detail

```
singleEventListener = new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot)
{
        key = dataSnapshot.getKey().toString();
        ModelFleet fleet =
dataSnapshot.getValue(ModelFleet.class);
        ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.cla
ss);
        Geocoder geocoder; List<Address> _address;
        geocoder = new
Geocoder(getApplicationContext(), Locale.getDefault());
        try {
            _address =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
            address =
_address.get(0).getAddressLine(0);
        } catch (IOException e) {
            e.printStackTrace();
        }

        tv_ID_Fill.setText(fleet.getEmail());
        txt_full_name.setText(fleet.getFullname());
        txt_email.setText(fleet.getEmail());
        txt_password.setText("yourpassword");
        txt_licence.setText(fleet.getLicense());
        txt_phone.setText(fleet.getPhone());
        txt_address.setText(address);
        txt_gender.setText(fleet.getGender());
        txt_birthdate.setText(fleet.getBirthdate());
        txt_truckname.setText(fleet.getTruckName());
        txt_truckplate.setText(fleet.getTruckPlate());
    }

    @Override
    public void onCancelled(DatabaseError
databaseError) {

    }
};
new
FireDataUserFleet().ref.child(message).addListenerForSi
ngleValueEvent(singleEventListener);
```

Gambar 5.33 US-AM1-5: Mengambil data armada

Untuk mengedit masing-masing data, maka *user* harus menekan ikon pensil. Pada *email* dan *password*, *user* tidak bisa

mengeditnya karena hanya *user* yang memiliki data langsung yang dapat mengubahnya. Berikut potongan kode untuk mengedit masing-masing data:

1. Nama lengkap
   Untuk mengedit nama lengkap sopir, langkahnya adalah mengambil teks yang ada dalam *EditText* lalu menyimpannya dalam databsae, menggunakan kode pada gambar 5.34.

```
@OnClick(R.id.btn_save_driver_fullname) public void
saveName(){
    newName =
et_edit_fullname.getText().toString();

    if(TextUtils.isEmpty(newName)){
        Toast.makeText(this, "new Name is empty",
Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserFleet().writeUserFullName(message,
newName, new DatabaseReference.CompletionListener()
{
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Fleet_Edit_Name.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.34 US-AM1-5: Edit nama lengkap sopir Linsensi SIM

2. Lisensi SIM

   Untuk mengedit lisensi sopir, langkahnya adalah mengambil teks yang ada dalam *EditText* lalu menyimpannya dalam *database*, menggunakan kode pada gambar 5.35.

```
@OnClick(R.id.btn_save_driver_license) public void
saveLicense(){
    newLicense =
et_edit_license.getText().toString();

    if(TextUtils.isEmpty(newLicense)){
        Toast.makeText(this, "new driver license is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserFleet().writeUserLicense(message,
newLicense, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Fleet_Edit_DriverLicense.this,
"Error while updating : " +
databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.35 US-AM1-5: Edit nomor lisensi SIM sopir

3. Nomor telepon

   Untuk mengedit nomor telepon, langkahnya adalah mengambil teks yang ada dalam *EditText* lalu menyimpannya dalam *database*. Namun belum ada aturan

yang melarang *user* saat mengisikan teks yang tidak sesuai dengan format nomor telepon. Kode yang digunakan adalah pada gambar 5.36.

```
@OnClick(R.id.btn_save_driver_phonenumber) public
void savePhone(){
    newPhone =
et_edit_phonenumber.getText().toString();

    if(TextUtils.isEmpty(newPhone)){
        Toast.makeText(this, "new phone number is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserFleet().writeUserPhone(message,
newPhone, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Fleet_Edit_Phone.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.36 US-AM1-5: Edit nomor telepon sopir

4. Alamat lengkap

   Untuk mengedit alamat lengkap, langkahnya adalah mengambil kordinat lokasi menggunakan akticitas Account_Set_Map. Setelah didapatkan lalu menyimpannya ke dalam *database* menggunakan kode pada gambar 5.37.

```
@Override protected void onActivityResult(int
requestCode, int resultCode, Intent data){
    if(requestCode==3){
        if(resultCode == Activity.RESULT_OK){
            latitude =
Double.parseDouble(data.getStringExtra("latitude"))
;
            longitude =
Double.parseDouble(data.getStringExtra("longitude")
);
            latLng = new LatLng(latitude,
longitude);

            Geocoder geocoder;
            List<Address> addresses;
            geocoder = new Geocoder(this,
Locale.getDefault());
            try {
                addresses =
geocoder.getFromLocation(latitude, longitude,1);
                address =
addresses.get(0).getAddressLine(0);
                et_edit_address.setText(address);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

@OnClick(R.id.btn_save_driver_address) public void
saveAddress(){
    address =
tv_toolbar_support.getText().toString();
    if(latLng==null){
        Toast.makeText(this, "Set address with a
map", Toast.LENGTH_SHORT).show();
        return;
    }
    if(TextUtils.isEmpty(address)){
        Toast.makeText(this, "new address is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserFleet().writeUserAddress(user.getUid(),
latLng, new DatabaseReference.CompletionListener()
{
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
```

```
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Fleet_Edit_Address.this, "Error
while updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.37 US-AM1-5: Edit alamat lengkap sopir

5. Jenis Kelamin
   Untuk megedit jenis kelamin, aplikasi mengambil teks yang ada dalam *EditText* yang berisi nilai "*female*" atau "*male*", lalu menyimpannya ke dalam *database*. Ada aturan yang melarang *user* saat mengisikan teks selain "*male*"atau "*female*" tersebut. Kode yang digunakan adalah pada gambar 5.38.
6. Tanggal lahir
   Untuk mengedit tanggal lahir, langkahnya adalah mengambil teks yang ada dalam *EditText* dengan format (MM/DD/YYYY). Ada aturan yang melarang apabila format penulisan tanggal salah. Kode yang digunakan adalah pada gambar 5.39.
7. Nama truk
   Untuk mengedit nama truk, langkahnya adalah mengambil teks yang ada dalam *EditText*, lalu menyimpannya dalam *database* menggunakan kode pada gambar 5.40
8. Nomor plat truk
   Untuk mengedit nomor plat truk, langkahnya adalah mengambil teks yang ada dalam *EditText* setelah *user* mengisikan nomor platnya. Namun belum ada aturan yang melarang *user* saat mengisikan teks dengan nomor plat

yang sesuai aturan. Kode yang digunakan adalah pada gambar 5.41.

```
@OnClick(R.id.btn_save_driver_gender) public void
saveGender(){
    newGender = et_edit_gender.getText().toString();

    String gender = newGender.trim().toLowerCase();
    if(gender.equals("male") || gender.equals("female"))
{
        if(TextUtils.isEmpty(gender)){
            Toast.makeText(this, "new gender is empty",
Toast.LENGTH_SHORT).show();
            return;
        }
        else{
            new
FireDataUserFleet().writeUserGender(message, gender, new
DatabaseReference.CompletionListener() {
                @Override
                public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                    if(databaseError == null){
                        finish();
                    }
                    else{
Toast.makeText(Fleet_Edit_Gender.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    }
    else{
        Toast.makeText(this, "input format invalid",
Toast.LENGTH_SHORT).show();
    }
}
```

Gambar 5.38 US-AM1-5: Edit jenis kelamin sopir

```
private boolean validate(String date){
    String expression = "(0?[1-9]|1[012])+[/.-
]+(0?[1-9]|[12][0-9]|3[01])+[/.-]+((19|20)\\d\\d)";
    Matcher matcher =
Pattern.compile(expression).matcher(date);

    if(matcher.matches()){
        String value = matcher.group(0);
        int month =
Integer.parseInt(value.substring(0,2));
        int day =
Integer.parseInt(value.substring(3,5));
        int year =
Integer.parseInt(value.substring(6));
        Log.d("tsss","Value: "+ month + " : " + day
+ " : " + year);

        if ((day<0 || day>30) && (month==4 ||
month==6 || month==9 ||month==11)) {

            Log.d("tsss", "birthdate check: false
month 4,6,9,11");
            return false;
        }
        else if((day<0 || day>31) && (month==1 ||
month==3 || month==5 ||month==7 ||month==8
||month==10 ||month==12)){
            Log.d("tsss", "birthdate check: false
month 1,3,5,7,8,10,12");
            return false;
        }
        else if (month==2) {
            if(year % 4==0){
                if(day<0 || day>29){
                    Log.d("tsss", "birthdate check:
false month 2 date 30,31");
                    return false;
                }
                else{
                    return true;
                }
            }
            else{
                if(day>28){
                    Log.d("tsss", "birthdate check:
false month 2 date 29,30,31");
                    return false;
                }
                else{
                    return true;
                }
            }
```

```
        }
        else if(month<0 || month >12){
            return false;
        }
        else{
            return true;
        }
    }
    else{
        Log.d("tsss", "matcher no match");
        return false;
    }
}

@OnClick(R.id.btn_save_driver_birthday) public void
saveBirthdate(){
    newBirthDate =
et_edit_birthdate.getText().toString();

    if(TextUtils.isEmpty(newBirthDate)){
        Toast.makeText(this, "new birth date is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else if(newBirthDate.length()!=10){
        Toast.makeText(Fleet_Edit_BirthDate.this,
"input format invalid", Toast.LENGTH_SHORT).show();
    }
    else{
        if(validate(newBirthDate)){
            new
FireDataUserFleet().writeUserBirthDate(message,
newBirthDate, new
DatabaseReference.CompletionListener() {
                @Override
                public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                    if(databaseError == null){
                        finish();
                    }
                    else{

Toast.makeText(Fleet_Edit_BirthDate.this, "Error
while updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
        else{
```

```
Toast.makeText(Fleet_Edit_BirthDate.this, "input
format invalid", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Gambar 5.39 US-AM1-5: Edit tanggal lahir sopir

```
@OnClick(R.id.btn_save_driver_vehiclename)
public void saveName(){
    newFleetName =
et_edit_vehiclename.getText().toString();

    if(TextUtils.isEmpty(newFleetName)){
        Toast.makeText(this, "new vehicle
name is empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserFleet().writeUserTruckName(messag
e, newFleetName, new
DatabaseReference.CompletionListener() {
            @Override
            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Fleet_Edit_TruckName.this,
"Error while updating : " +
databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

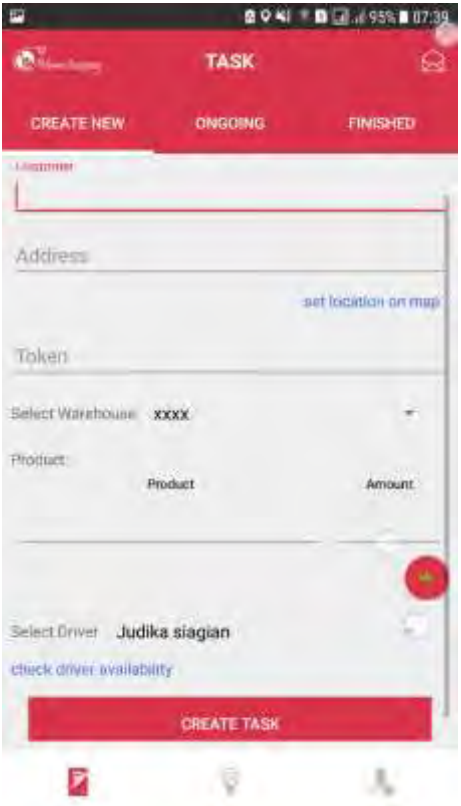
Gambar 5.40 US-AM1-5: Edit nama truk

```
@OnClick(R.id.btn_save_driver_licenseplate) public
void savePlate(){
    newPlate =
et_edit_licenseplate.getText().toString();

    if(TextUtils.isEmpty(newPlate)){
        Toast.makeText(this, "new vehichle plate
number is empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUserFleet().writeUserTruckPlate(message,
newPlate, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(Fleet_Edit_Plate.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.41 US-AM1-5: US-AM1-5: Edit nomor plat truk

5.2.2.6 US-AM2-1: Membuat penugasan baru

Untuk membuat penugasan baru, *activity* yang digunakan adalah **New_Create**. Aktivitas tersebut memiliki tampilan pada gambar 5.42. Activity ini menggunakan 2 *spinner* untuk menampilkan daftar sopir dan daftar gudang yang bisa digunakan oleh perusahaan, sehingga *activity* ini menggunakan implementasi pada gambar 5.43.



Gambar 5.42 US-AM2-1: Tampilan New_Create

```
public class New_Create extends Fragment
implements OnItemSelectedListener {

}
```

Gambar 5.43 US-AM2-1: Implements onItemSelectedListener

Sebelum bisa memilih data dari *spinner*, data harus diambil dari *Firebase* dimasukkan ke dalam *list* yang nantinya akan digunakan *spinner*. Kode yang digunakan untuk mengambil

data *Firebase* pada gambar 5.44 untuk mengambil data Sopir dan gambar 5.45 untuk mengambil data gudang.

```
private void prepareDriverList(){
    if(driverAvailabilityList.size() !=
0)driverAvailabilityList.clear();
    if(driverAvailabilityList1.size() !=
0)driverAvailabilityList1.clear();
    driverEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {
            ModelFleet _fleet =
dataSnapshot.getValue(ModelFleet.class);
            String isSignUp =
dataSnapshot.child(FireDataUserFleet.IS_SIGNUP).getV
alue().toString();
            String tmpAdmin =
dataSnapshot.child(FireDataUserFleet.COMPANY_ID).get
Value().toString();
            ModelFleet fleet = new
ModelFleet(dataSnapshot.getKey(),
_fleet.getFullname());
            if(companyID.equals(tmpAdmin)){
                if(isSignUp.equals("true")){

driverAvailabilityList.add(fleet);

driverAvailabilityList1.add(fleet.getFullname());
                }
            }


            ArrayAdapter<String> dataAdapter = new
ArrayAdapter<>(getContext(),

android.R.layout.simple_spinner_dropdown_item,
driverAvailabilityList1);

dataAdapter.setDropDownViewResource(android.R.layout
.simple_spinner_dropdown_item);
            spin_driver.setAdapter(dataAdapter);
        }

        @Override
        public void onChildChanged(DataSnapshot
dataSnapshot, String s) {
```

```
            ModelFleet _fleet =
dataSnapshot.getValue(ModelFleet.class);
            String isSignUp =
dataSnapshot.child(FireDataUserFleet.IS_SIGNUP).getV
alue().toString();
            String tmpAdmin =
dataSnapshot.child(FireDataUserFleet.COMPANY_ID).get
Value().toString();
            ModelFleet fleet = new
ModelFleet(dataSnapshot.getKey(),
_fleet.getFullname());
            if(companyID.equals(tmpAdmin)){
                if(isSignUp.equals("true")){


if(driverAvailabilityList.contains(fleet)){

driverAvailabilityList.remove(fleet);
                    }

if(driverAvailabilityList1.contains(fleet.getFullnam
e())){

driverAvailabilityList1.remove(fleet.getFullname());
                    }


driverAvailabilityList.add(fleet);

driverAvailabilityList1.add(fleet.getFullname());
                }
            }

            ArrayAdapter<String> dataAdapter = new
ArrayAdapter<>(getContext(),

android.R.layout.simple_spinner_dropdown_item,
driverAvailabilityList1);

dataAdapter.setDropDownViewResource(android.R.layout
.simple_spinner_dropdown_item);
            spin_driver.setAdapter(dataAdapter);
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            ModelFleet _fleet =
```

```
dataSnapshot.getValue(ModelFleet.class);
            ModelFleet fleet = new
ModelFleet(dataSnapshot.getKey(),
_fleet.getFullname());
            driverAvailabilityList.remove(fleet);

driverAvailabilityList1.remove(fleet.getFullname());

            ArrayAdapter<String> dataAdapter = new
ArrayAdapter<>(getContext(),

android.R.layout.simple_spinner_dropdown_item,
driverAvailabilityList1);

dataAdapter.setDropDownViewResource(android.R.layout
.simple_spinner_dropdown_item);
            spin_driver.setAdapter(dataAdapter);
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataUserFleet().ref.addChildEventListener(driver
EventListener);
    spin_driver.setOnItemSelectedListener(this);
}
```

Gambar 5.44 US-AM2-1: Mengambil data sopir dari Firebase

```
private void prepareWareHouseList(){
    wareHouseEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            warehouseList.clear();
            warehouseList1.clear();
            for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                ModelWarehouse _modelWarehouse =
postSnapshot.getValue(ModelWarehouse.class);
                if(_modelWarehouse != null){

if(_modelWarehouse.getCompanyID().equals(user.getUid())
){
                        ModelAddress modelAddress =
postSnapshot.child("address").getValue(ModelAddress.cla
ss);
                        LatLng latlngx = new
LatLng(modelAddress.getLatitude(),
modelAddress.getLongitude());
                        ModelWarehouse modelWarehouse =
new ModelWarehouse(postSnapshot.getKey(),
_modelWarehouse.getName(), latlngx);


warehouseList.add(modelWarehouse);

warehouseList1.add(modelWarehouse.getName());

                        ArrayAdapter<String>
dataAdapter = new
ArrayAdapter<>(getActivity().getApplicationContext(),

android.R.layout.simple_spinner_dropdown_item,
warehouseList1);

dataAdapter.setDropDownViewResource(android.R.layout.si
mple_spinner_dropdown_item);

spin_wareHouse.setAdapter(dataAdapter);
                        Log.d("tsss", "warehouse from
firebase: "+modelWarehouse.getName());
                    }
                }
            }
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {
```

```
        }
    };
    new
FireDataWareHouse().ref.addValueEventListener(wareHouse
EventListener);
    spin_wareHouse.setOnItemSelectedListener(this);
}
```

Gambar 5.45 US-AM2-1: Mengambil data gudang dari Firebase

Data sopir menggunakan *ChildEventListener* karena data sopir bisa saja berubah-ubah saat sedang membuat *task*, namun data gudang tidak bisa diubah karena yang bisa mengubahnya hanya 1 *user* sehingga data gudang menggunakan *ValueEventListener*. Data sopir dimasukkan ke dalam *ArrayList driverAvalabilityList1* yang nantinya akan digunakan ke spin_driver (*spinner* untuk menampilkan *list* sopir), Data *warehouse* dimasukkan ke dalam *ArrayList warehouseList1* yang nantinya akan dimasukkan ke spin_wareHouse (*spinner* untuk menampilkan *list* gudang). Untuk mengambil data dari pilihan *spinner* digunakan kode pada gambar 5.46.

```
@Override
public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
    Spinner spinner = (Spinner) parent;
    if(spinner.getId() == R.id.spin_driver){
        spin_driver.setSelection(position);
        fleetID =
driverAvailabilityList.get(position).getID();
    }
    if(spinner.getId() == R.id.spin_wareHouse){
        spin_wareHouse.setSelection(position);
        wareHouseName =
warehouseList.get(position).getName();
        wareHouseLoc =
warehouseList.get(position).getLatLng();
    }

}
```

Gambar 5.46 US-AM2-1: onItemSelectedListener

List barang yang akan diantar diketikkan oleh *user*. Tiap barang diisikan dalam 1 *card* yang dimuat dalam *LinearLayout* ll_product yang letaknya diatas *FloatingButton* btn_add atau pada gambar 5.24 berbentuk garis biru horizontal panjang. Pada awal aktivitias dimulai, *card* hanya ada 1 untuk mengisikan 1 produk, jika ingin menambahkan lebih banyak produk, makan harus menekan btn_add yang akan menjalankan fungsi card() dengan kode seperti gambar 5.47.

```
private void card(){
    listSize++;
    LayoutInflater _canvasProduct =
LayoutInflater.from(getContext());
    View canvasProduct =
_canvasProduct.inflate(R.layout.card_add_product,
null, false);
    EditText et_edit_productName = (EditText)
canvasProduct.findViewById(R.id.et_edit_productNa
me);
    EditText et_edit_productAmount = (EditText)
canvasProduct.findViewById(R.id.et_edit_productAm
ount);

    ModelMessage _message = new
ModelMessage(listSize, et_edit_productName,
et_edit_productAmount);
    messageList.add(_message);

    ll_product.addView(canvasProduct);
    ll_product.requestLayout();
}
```

Gambar 5.47 US-AM2-1: Menambah Card Produk

Setelah semua data penugasan diisikan, maka kode yang dijalankan adalah pada gambar 5.48.

```
public void createTask(){
    customer = et_edit_customer.getText().toString();
    token = et_edit_token.getText().toString();
    if(TextUtils.isEmpty(customer)){
        Toast.makeText(getContext(), "Please enter your
customer name", Toast.LENGTH_SHORT).show();
        return;
    }
    else if(TextUtils.isEmpty(token)){
        Toast.makeText(getContext(), "Please enter
TOKEN", Toast.LENGTH_SHORT).show();
        return;

    else if(latLng==null){
        Toast.makeText(getContext(), "Please choose
location with map", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        for(int i = 0; i< messageList.size(); i++){
            String id = messageList.get(i).getId() +
"";
            String item =
messageList.get(i).getEtItem().getText() + "";
            String amount =
messageList.get(i).getEtTarget().getText() + "";

            Map<String, Object> objectMessage = new
HashMap<>();

objectMessage.put(FireDataTask.MESSAGE_VALUE_ITEM,
item);

objectMessage.put(FireDataTask.MESSAGE_VALUE_TARGET,
amount);

            messageMap.put(id, objectMessage);
            if(i == messageList.size()-1){
                new
FireDataTask().writeNewTask(companyID, wareHouseName,
wareHouseLoc, fleetID, latLng, customer, token,
messageMap,
                        new
DatabaseReference.CompletionListener(){
                            @Override
                            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {

Toast.makeText(getActivity(),"Task succesfully
created", Toast.LENGTH_SHORT).show();
```

```
et_edit_customer.setText("");

et_edit_address.setText("");

et_edit_token.setText("");
                                messageList.clear();
                                messageMap.clear();
                                listSize=0;

ll_product.removeAllViews();
                                card();
                            }
                        });
                }
            }
        }
}
```

Gambar 5.48 US-AM2-1: Membuat penugasan baru

5.2.2.7 US-AM2-2: Melihat lokasi dan tugas sopir

Melihat lokasi dan tugas sopir ditampilkan dalam 2 *fragment* yang berbeda yakni *FleetStatusList* untuk menampilkan informasi armada dalam bentuk daftar berisi teks, dan *FleetStatusMap* untuk menampilkan informasi lokasi armada, lokasi perusahaan, dan lokasi-lokasi gudang dalam visualisasi peta dengan pin-pin yang dikustomisasi.

1. *FleetStatusList*
   Pada *fragment* ini, data yang ditampilkan hanya tentang armada dengan informasi yang ditampilkan yakni: nama sopir, bekerja/tidak bekerja, dan lokasi terkini sopir. Semua data ini tersedia pada 1 ranting *Fleet* di *Firebase* jadi hanya perlu mengambil 1 data saja. Pengambilan data menggunakan fungsi pada gambar 5.49.

```
private void initFleetList(){
    fleetEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {

if(adminKey.equals(dataSnapshot.child("companyID").
getValue().toString())){

if(dataSnapshot.child("isSignUp").getValue().toStri
ng().equals("true")){
                    String key =
dataSnapshot.getKey();
                    ModelFleet fleet =
dataSnapshot.getValue(ModelFleet.class);
                    ModelAddress modelAddress =
dataSnapshot.child("location").getValue(ModelAddres
s.class);
                    if(modelAddress != null){
                        String location=null;
                        String driverName =
fleet.getFullname();
                        boolean isWorking =
fleet.isWorking();
                        String workingStatus;
                        if(isWorking==true)
workingStatus = "delivering order";
                        else workingStatus =
"idling/available";

                        Geocoder geocoder;
                        List<Address> _address;
                        geocoder = new
Geocoder(getContext(), Locale.getDefault());
                        try {
                            _address =
geocoder.getFromLocation(modelAddress.getLatitude()
, modelAddress.getLongitude(),1);
                            location =
_address.get(0).getAddressLine(0);
                        } catch (IOException e) {
                            e.printStackTrace();
                        }

                        fleetListMap.put(key, new
ModelFleetList(driverName,workingStatus,location));

mAdapter.notifyDataSetChanged();
                    }
                    checkEmpty();
                }
            }
```

```
        }

        @Override
        public void onChildChanged(DataSnapshot
dataSnapshot, String s) {

if(adminKey.equals(dataSnapshot.child("companyID").
getValue().toString())){

if(dataSnapshot.child("isSignUp").getValue().toStri
ng().equals("true")){
                    String key =
dataSnapshot.getKey();
                    ModelFleet fleet =
dataSnapshot.getValue(ModelFleet.class);
                    ModelAddress modelAddress =
dataSnapshot.child("location").getValue(ModelAddres
s.class);
                    if(modelAddress != null){
                        String location=null;
                        String driverName =
fleet.getFullname();
                        boolean isWorking =
fleet.isWorking();
                        String workingStatus;
                        if(isWorking==true)
workingStatus = "delivering order";
                        else workingStatus =
"idling/available";

                        Geocoder geocoder;
                        List<Address> _address;
                        geocoder = new
Geocoder(getContext(), Locale.getDefault());
                        try {
                            _address =
geocoder.getFromLocation(modelAddress.getLatitude()
, modelAddress.getLongitude(),1);
                            location =
_address.get(0).getAddressLine(0);
                        } catch (IOException e) {
                            e.printStackTrace();
                        }

                        fleetListMap.put(key, new
ModelFleetList(driverName,workingStatus,location));

mAdapter.notifyDataSetChanged();
                    }

                    checkEmpty();
                }
```

```
            }
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            String key = dataSnapshot.getKey();
            fleetListMap.remove(key);
            mAdapter.notifyDataSetChanged();
            checkEmpty();
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataUserFleet().ref.addChildEventListener(fleet
EventListener);
}
```

Gambar 5.49 US-AM2-2: FleetStatusList-Mengambil data armada

Data tiap armada dimasukkan ke dalam *HashMap FleetListMap*. *HashMap* ini nantinya dimasukkan ke dalam *adapter* yang memungkinkan 1 key pada *HashMap* ditampilkan dalam 1 *card*. *Adapter* akan ditampilkan dalam *RecyclerView* recyclerview_Fleet. Apabila tidak ada armada yang terdata, *RecyclerView* tidak muncul, namun memunculkan *TextView* yang menjelaskan tidak ada armada terdata. Kode yang digunakan adalah pada gambar 5.50.

```
private void initRecycler(){
    mAdapter = new
FleetListAdapter(fleetListMap);
    RecyclerView.LayoutManager mLayoutManager =
new LinearLayoutManager(getActivity());

recyclerview_fleet.setLayoutManager(mLayoutManag
er);
    recyclerview_fleet.setItemAnimator(new
DefaultItemAnimator());
    recyclerview_fleet.setAdapter(mAdapter);

    checkEmpty();
}

private void checkEmpty(){
    if(fleetListMap.size()>0){

recyclerview_fleet.setVisibility(View.VISIBLE);
        tv_emptyFleet.setVisibility(View.GONE);
    }
    else {

recyclerview_fleet.setVisibility(View.GONE);

tv_emptyFleet.setVisibility(View.VISIBLE);
    }
}
```

Gambar 5.50 US-AM2-2: set Adapter pada recycler view

Konfigurasi pada *Adapter* adalah pada gambar 5.51. Kemudian Adapter tersebut menggunakan card_Fleet_list dengan tampilan pada gambar 5.52.

```
private Map<String, ModelFleetList> fleetList;

public class MyViewHolder extends
RecyclerView.ViewHolder{

    @BindView(R.id.tv_driverName) TextView
tv_driverName;
    @BindView(R.id.tv_workingStatus) TextView
tv_workingStatus;
    @BindView(R.id.tv_location) TextView tv_location;
    @BindView(R.id.rlCard) LinearLayout rlCard;

    public MyViewHolder(View itemView) {
        super(itemView);
        ButterKnife.bind(this, itemView);
    }
}

public FleetListAdapter(Map<String, ModelFleetList>
fleetList){
    this.fleetList = fleetList;
}

@Override
public MyViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
    View view =
LayoutInflater.from(parent.getContext()).
            inflate(R.layout.card_fleet_list,parent,
false);

    return new MyViewHolder(view);
}

@Override
public void
onBindViewHolder(FleetListAdapter.MyViewHolder holder,
int position) {
    final List<String> keyList = new
ArrayList<>(fleetList.keySet());
    final List<ModelFleetList> fleetLists = new
ArrayList<>(fleetList.values());
    ModelFleetList fleet = fleetLists.get(position);

    holder.tv_driverName.setText(fleet.getFullname());

holder.tv_workingStatus.setText(fleet.getWorkingStatus(
));
    holder.tv_location.setText(fleet.getLocation());
}

@Override
```

```
public int getItemCount() {
    return fleetList.size();
}
```

Gambar 5.51 US-AM2-2: FleetListAdapter

Gambar 5.52 US-AM2-2: card_Fleet_list

2. *FleetStatusMap*

   Fragment ini memiliki 4 jenis pin dan memiliki tombol *recenter* yang memindahkan kamera peta ke lokasi perusahaan. Pin tersebut adalah pin lokasi perusahaan, pin gudang, pin armada yang sedang bekerja, dan pin armada yang tidak bekerja. Untuk menampilkan pin tersebut, ada 3 cabang *Firebase* yang harus didapatkan dari *Firebase*

yakni *user*s, *Fleet*, dan *warehouse*. Berikut adalah *user interface FleetStatusMap*, pada gambar 5.53.

Gambar 5.53 US-AM2-2: Tampilan FleetStatusMap

Untuk mengambil lokasi perusahaan, digunakan kode pada gambar 5.54. Setelah kordinat perusahaan didapatkan, dibuat *marker* bernama *companyMarker* pada peta. Kordinat perusahaan juga sudah disimpan pada variable *LatLng latLngCompany*

Untuk mengambil data armada, digunakan kode pada gambar 5.55. Setelah semua kordinat armada didapatkan, dibuat *marker*

yang bernama *fleetMarker*. Kordinat armada disimpan pada variable *ArrayList*<LatLng> *FleetList*. Saat ada perubahan data 1 armada, semua *marker* dihapus, *FleetList* dihapus, *warehouseMarker* ditampilkan lagi dari *warehouseList*, dan *companyMarker* dari *latLngCompany*. Data dari *Firebase* diambil lagi dan menyimpan datanya di *FleetList* sekaligus menampilkan *fleetMarker*.

```
private void setCompanyLocation(){
    if(mMap != null) mMap.clear();
    companyEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            ModelAddress address =
dataSnapshot.child(FireDataUserCompany.ADDRESS).getValu
e(ModelAddress.class);
            companyLatitude = address.getLatitude();
            companyLongitude = address.getLongitude();
            latLngCompany = new LatLng(companyLatitude,
companyLongitude);

            companyMarker= mMap.addMarker(new
MarkerOptions().position(latLngCompany).title("Company"
).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_company", 52, 100))));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLn
gCompany, 12));
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataUserCompany().ref.child(companyID).addListenerF
orSingleValueEvent(companyEventListener);
}
```

Gambar 5.54 US-AM2-2: Mengambil data company dari Firebase

```
private void setFleetLocation(){
    fleetEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {
            ModelFleet fleet =
dataSnapshot.getValue(ModelFleet.class);

if(companyID.equals(fleet.getcompanyID())){

if(dataSnapshot.child("isSignUp").getValue().toString
().equals("true")){
                    String fullname =
fleet.getFullname();
                    boolean isWorking =
fleet.isWorking();
                    ModelAddress modelAddress =
dataSnapshot.child("location").getValue(ModelAddress.
class);
                    if(modelAddress !=null){
                        ModelFleet modelFleet = new
ModelFleet(isWorking,fullname,
modelAddress.getLatitude(),
modelAddress.getLongitude());
                        fleetlist.add(modelFleet);

                        for(int i=0;
i<fleetlist.size();i++){
                            String tempFleetName =
fleetlist.get(i).getFullname();
                            LatLng tempFleetLatLng =
new LatLng(fleetlist.get(i).getLat(),
fleetlist.get(i).getLng());
                            boolean tempIsWorking =
fleetlist.get(i).isWorking();
                            if(tempIsWorking==true){
                                fleetMarker =
mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempF
leetName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_fleet_current_location_2", 47, 80))));
                            }
                            else{
                                fleetMarker =
mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempF
leetName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_fleet_current_location", 47, 80))));
```

```
                            }
                        }
                    }
                }
            }
        }

        @Override
        public void onChildChanged(DataSnapshot
dataSnapshot, String s) {
            mMap.clear();
            fleetlist.clear();
            //return warehouse marker
            for(int i=0; i<wareHouseList.size();i++){
                String tempWareHouseName =
wareHouseList.get(i).getName();
                LatLng tempWareHouseLatLng =
wareHouseList.get(i).getLatLng();
                wareHouseMarker = mMap.addMarker(new
MarkerOptions().position(tempWareHouseLatLng).title(t
empWareHouseName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_warehouse", 42, 80))));
            }

            //return company marker
            companyMarker= mMap.addMarker(new
MarkerOptions().position(latLngCompany).title("Compan
y").

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_company", 52, 100))));

            //return fleet marker
            fleetEventListener2 = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot2) {
                    for(DataSnapshot postSnapshot :
dataSnapshot2.getChildren()){

if(companyID.equals(postSnapshot.child("companyID").g
etValue().toString())) {
                            if
(postSnapshot.child("isSignUp").getValue().toString()
.equals("true")) {
                                ModelFleet fleet =
postSnapshot.getValue(ModelFleet.class);
                                String fullname =
fleet.getFullname();
```

```
                                boolean isWorking =
fleet.isWorking();
                                ModelAddress
modelAddress =
postSnapshot.child("location").getValue(ModelAddress.
class);
                                if(modelAddress
!=null){
                                    ModelFleet
modelFleet = new ModelFleet(isWorking,fullname,
modelAddress.getLatitude(),
modelAddress.getLongitude());

fleetlist.add(modelFleet);

                                for(int i=0;
i<fleetlist.size();i++){
                                    String
tempFleetName =  fleetlist.get(i).getFullname();
                                    LatLng
tempFleetLatLng = new
LatLng(fleetlist.get(i).getLat(),
fleetlist.get(i).getLng());
                                    boolean
tempIsWorking = fleetlist.get(i).isWorking();

if(tempIsWorking==true){

fleetMarker = mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempF
leetName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_fleet_current_location_2", 47, 80))));
                                    }
                                    else{

fleetMarker = mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempF
leetName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_fleet_current_location", 47, 80))));
                                    }
                                }
                            }
                        }
                    }
                }

                @Override
```

```
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataUserFleet().ref.addValueEventListener(fleetEv
entListener2);
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            mMap.clear();
            fleetlist.clear();
            //return warehouse marker
            for(int i=0; i<wareHouseList.size();i++){
                String tempWareHouseName =
wareHouseList.get(i).getName();
                LatLng tempWareHouseLatLng =
wareHouseList.get(i).getLatLng();
                wareHouseMarker = mMap.addMarker(new
MarkerOptions().position(tempWareHouseLatLng).title(t
empWareHouseName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_warehouse", 42, 80))));
            }

            //return company marker
            companyMarker= mMap.addMarker(new
MarkerOptions().position(latLngCompany).title("Compan
y").

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_company", 52, 100))));

            //return fleet marker
            fleetEventListener2 = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot2) {
                    for(DataSnapshot postSnapshot :
dataSnapshot2.getChildren()){

if(companyID.equals(postSnapshot.child("companyID").g
etValue().toString())) {
                            if
(postSnapshot.child("isSignUp").getValue().toString()
.equals("true")) {
                                ModelFleet fleet =
```

```
postSnapshot.getValue(ModelFleet.class);
                                String fullname =
fleet.getFullname();
                                boolean isWorking =
fleet.isWorking();
                                ModelAddress
modelAddress =
postSnapshot.child("location").getValue(ModelAddress.
class);
                                if(modelAddress
!=null){
                                    ModelFleet
modelFleet = new ModelFleet(isWorking,fullname,
modelAddress.getLatitude(),
modelAddress.getLongitude());

fleetlist.add(modelFleet);

                                for(int i=0;
i<fleetlist.size();i++){
                                    String
tempFleetName =  fleetlist.get(i).getFullname();
                                    LatLng
tempFleetLatLng = new
LatLng(fleetlist.get(i).getLat(),
fleetlist.get(i).getLng());
                                    boolean
tempIsWorking = fleetlist.get(i).isWorking();

if(tempIsWorking==true){

fleetMarker = mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempF
leetName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_fleet_current_location_2", 47, 80))));
                                    }
                                    else{

fleetMarker = mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempF
leetName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap(
"map_fleet_current_location", 47, 80))));
                                    }
                                }
                            }
                        }
                    }
```

```
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataUserFleet().ref.addValueEventListener(fleetEv
entListener2);
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataUserFleet().ref.addChildEventListener(fleetEv
entListener);
}
```

Gambar 5.55 US-AM2-2: Mengambil data armada dari Firebase

Untuk mengambil data gudang, digunakan kode pada gambar 5.56. Setelah semua kordinat gudang didapatkan, dibuat *marker* yang bernama *warehouseMarker* pada peta. Kordinat armada juga disimpan pada variable *ArrayList*<LatLng> *warehouseList*. Saat ada perubahan data 1 gudang, semua *marker* yang ada di googlemap dihapus, *warehouseList* dihapus, *FleetMarker* ditampilkan lagi dari *FleetList* yang, *companyMarker* ditampilkan lagi dari *latLngCompany*. Untuk mengisi *warehouseList*, maka data dari *Firebase* diambil lagi dan menyimpan datanya di *warehouseList* sekaligus menampilkan *warehouseMarker*.

```
private void getWareHouseData(){
    wareHouseEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {
            ModelWarehouse _modelWarehouse =
dataSnapshot.getValue(ModelWarehouse.class);

if(companyID.equals(_modelWarehouse.getCompanyID())){
                ModelAddress addressWareHouse =
dataSnapshot.child(FireDataWareHouse.ADDRESS).getValue(
ModelAddress.class);
                String wareHouseName =
_modelWarehouse.getName();
                LatLng _temp = new
LatLng(addressWareHouse.getLatitude(),
addressWareHouse.getLongitude());
                ModelWarehouse modelWarehouse = new
ModelWarehouse(dataSnapshot.getKey(),wareHouseName,
_temp);
                wareHouseList.add(modelWarehouse);

                for(int i=0;
i<wareHouseList.size();i++){
                    String tempWareHouseName =
wareHouseList.get(i).getName();
                    LatLng tempWareHouseLatLng =
wareHouseList.get(i).getLatLng();
                    wareHouseMarker =
mMap.addMarker(new
MarkerOptions().position(tempWareHouseLatLng).title(tem
pWareHouseName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_warehouse", 42, 80))));
                }
            }
        }

        @Override
        public void onChildChanged(DataSnapshot
dataSnapshot, String s) {
            wareHouseList.clear();
            mMap.clear();

            //return company marker
            companyMarker= mMap.addMarker(new
MarkerOptions().position(latLngCompany).title("Company"
).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_company", 52, 100))));
```

```
            //return fleet marker
            for(int i=0; i<fleetlist.size();i++){
                String tempFleetName =
fleetlist.get(i).getFullname();
                LatLng tempFleetLatLng = new
LatLng(fleetlist.get(i).getLat(),
fleetlist.get(i).getLng());
                boolean tempIsWorking =
fleetlist.get(i).isWorking();
                if(tempIsWorking==true){
                    fleetMarker = mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempFle
etName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_fleet_current_location_2", 47, 80))));
                }
                else{
                    fleetMarker = mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempFle
etName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_fleet_current_location", 47, 80))));
                }
            }

            //return warehouse marker
            wareHouseEventListener2 = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot) {
                    for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                        ModelWarehouse _modelWarehouse
= postSnapshot.getValue(ModelWarehouse.class);

if(companyID.equals(_modelWarehouse.getCompanyID())){
                            ModelAddress
addressWareHouse =
postSnapshot.child(FireDataWareHouse.ADDRESS).getValue(
ModelAddress.class);
                            String wareHouseName =
_modelWarehouse.getName();
                            LatLng _temp = new
LatLng(addressWareHouse.getLatitude(),
addressWareHouse.getLongitude());
                            ModelWarehouse
modelWarehouse = new
ModelWarehouse(postSnapshot.getKey(),wareHouseName,
```

```
_temp);

wareHouseList.add(modelWarehouse);

                                for(int i=0;
i<wareHouseList.size();i++){
                                    String
tempWareHouseName =  wareHouseList.get(i).getName();
                                    LatLng
tempWareHouseLatLng = wareHouseList.get(i).getLatLng();
                                    wareHouseMarker =
mMap.addMarker(new
MarkerOptions().position(tempWareHouseLatLng).title(tem
pWareHouseName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_warehouse", 42, 80))));
                                }
                            }
                        }
                    }

                    @Override
                    public void onCancelled(DatabaseError
databaseError) {

                    }
                };
                new
FireDataWareHouse().ref.addValueEventListener(wareHouse
EventListener2);
            }

            @Override
            public void onChildRemoved(DataSnapshot
dataSnapshot) {
                wareHouseList.clear();
                mMap.clear();

                //return company marker
                companyMarker= mMap.addMarker(new
MarkerOptions().position(latLngCompany).title("Company"
).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_company", 52, 100))));

                //return fleet marker
                for(int i=0; i<fleetlist.size();i++){
                    String tempFleetName =
fleetlist.get(i).getFullname();
                    LatLng tempFleetLatLng = new
```

```
LatLng(fleetlist.get(i).getLat(),
fleetlist.get(i).getLng());
                boolean tempIsWorking =
fleetlist.get(i).isWorking();
                if(tempIsWorking==true){
                    fleetMarker = mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempFle
etName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_fleet_current_location_2", 47, 80))));
                }
                else{
                    fleetMarker = mMap.addMarker(new
MarkerOptions().position(tempFleetLatLng).title(tempFle
etName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_fleet_current_location", 47, 80))));
                }
            }

            //return warehouse marker
            wareHouseEventListener2 = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot) {
                    for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                        ModelWarehouse _modelWarehouse
= postSnapshot.getValue(ModelWarehouse.class);

if(companyID.equals(_modelWarehouse.getCompanyID())){
                            ModelAddress
addressWareHouse =
postSnapshot.child(FireDataWareHouse.ADDRESS).getValue(
ModelAddress.class);
                            String wareHouseName =
_modelWarehouse.getName();
                            LatLng _temp = new
LatLng(addressWareHouse.getLatitude(),
addressWareHouse.getLongitude());
                            ModelWarehouse
modelWarehouse = new
ModelWarehouse(postSnapshot.getKey(),wareHouseName,
_temp);

wareHouseList.add(modelWarehouse);

                            for(int i=0;
i<wareHouseList.size();i++){
```

```
                                String
tempWareHouseName =  wareHouseList.get(i).getName();
                                LatLng
tempWareHouseLatLng = wareHouseList.get(i).getLatLng();
                                wareHouseMarker =
mMap.addMarker(new
MarkerOptions().position(tempWareHouseLatLng).title(tem
pWareHouseName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_warehouse", 42, 80))));
                            }
                        }
                    }
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataWareHouse().ref.addValueEventListener(wareHouse
EventListener2);
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataWareHouse().ref.addChildEventListener(wareHouse
EventListener);
}
```

Gambar 5.56 US-AM2-2: Mengambil data gudang dari Firebase

5.2.2.8 US-AM2-3: Membatalkan pengiriman barang

Untuk membatalkan transaksi, maka *user* harus membuka detail dari penugasan, lalu menekan tombol "*cancel delivery*". Saat transaksi dibatalkan, penugasan harus segerea dihapus agar barang tidak perlu diantarkan ke pelanggan. Ada 2 kemungkinan yang bisa terjadi saat tugas dibatalkan sehingga perlakuan yang dibuat harus berbeda-beda agar tidak terjadi *error* pada *database* dan aplikasi. Kode untuk 2 kemungkinan tersebut adalah pada gambar 5.57

```
@OnClick(R.id.btn_cancel) public void cancelTask(){
    getActivity().onBackPressed();
    if(departureID==null){
        new
FireDataMessage().writeAbortTask(HomeActivity.companyUI
D, fleetID, taskID, customer, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                new
FireDataTask().ref.child(taskID).removeValue();
                getActivity().finish();
            }
        });
    }
    else{
        new
FireDataMessage().writeAbortTask(HomeActivity.companyUI
D, fleetID, taskID, customer, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                new
FireDataDeparture().ref.child(departureID).child("task"
).child(taskID).removeValue();
                new
FireDataTask().ref.child(taskID).removeValue();
                getActivity().finish();
            }
        });
    }
}
```

Gambar 5.57 US-AM2:3 Membatalkan pengiriman barang

1. Tugas belum diantarkan oleh armada
   Tugas yang belum diantarkan oleh armada ditandai dengan tidak adanya kode *departureID*. Saat tidak ada *departureID*, maka perintah yang dijalankan hanya memberikan notifikasi pada armada untuk tidak mengantarkan pesanan ke pelanggan bersangkutan, lalu menjalankan fungsi removeValue() pada cabang *taskID* yang bersangkutan.
2. Tugas sedang diantarkan oleh armada
   Tugas ini sudah memiliki kode *departureID*. Perintah yang dijalankan adalah memberikan notifikasi pada armada untuk tidak mengantarkan pesanan ke pelanggan yang bersangkutan. Lalu menghapus cabang dengan key *taskID* pada ranting *departure→departureID→task*, lalu menjalankan fungsi removeValue() pada cabang *taskID* yang bersangkutan.

5.2.2.9 US-AM2-4: Melihat penugasan yang sedang berjalan

Tugas-tugas yang sedang berjalan ditampilkan dalam *fragment Ongoing_List* yang disajikan dalam barisan *card*. Tiap *card* memiliki konten nama pelanggan, *departureID*, alamat pelanggan, dan tipe penugasan (*regular task* dan *rework task*). Semua data tersedia dalam 1 ranting *Task* di *Firebase* jadi hanya perlu mengambil 1 data saja. Data yang diambil hanya yang memiliki nilai ***false*** di ranting *Task→TaskID→isCompleted*. Pengambilan data *task* menggunakan pada gambar 5.58.

```
private void initTaskList(){
    childEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {

if(companyID.equals(dataSnapshot.child("companyID").
getValue().toString())){

if(dataSnapshot.child("isCompleted").getValue().toSt
ring().equals("false")){
                    final String key =
dataSnapshot.getKey();
                    final ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    final ModelAddress modelAddress
=
dataSnapshot.child("address").getValue(ModelAddress.
class);

                    singleEventListener = new
ValueEventListener() {
                        @Override
                        public void
onDataChange(DataSnapshot dataSnapshot) {
                            Geocoder geocoder;
                            List<Address> addresses;
                            geocoder = new
Geocoder(getContext(), Locale.getDefault());
                            try {
                                addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                                _address =
addresses.get(0).getAddressLine(0);
                            } catch (IOException e)
{
                                e.printStackTrace();
                            }

                            ModelFleet fleet =
dataSnapshot.getValue(ModelFleet.class);
                            String address =
_address;
                            String customer =
task.getCustomer();
                            String driverName =
fleet.getFullname();
```

```
                                String departureID =
task.getDepartureID();
                                String workType =
task.getWorkType();
                                OngoinglistMap.put(key,
new ModelOngoingList(customer, address, departureID,
driverName, workType));

mAdapter.notifyDataSetChanged();
                                checkEmpty();
                            }

                            @Override
                            public void
onCancelled(DatabaseError databaseError) {

                            }
                        };
                        new
FireDataUserFleet().ref.child(task.getFleetID()).add
ListenerForSingleValueEvent(singleEventListener);
                    }
                }
            }

            @Override
            public void onChildChanged(DataSnapshot
dataSnapshot, String s) {

if(companyID.equals(dataSnapshot.child("companyID").
getValue().toString())){

if(dataSnapshot.child("isCompleted").getValue().toSt
ring().equals("false")){
                        final String key =
dataSnapshot.getKey();
                        final ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                        final ModelAddress modelAddress
=
dataSnapshot.child("address").getValue(ModelAddress.
class);

                        singleEventListener = new
ValueEventListener() {
                            @Override
                            public void
onDataChange(DataSnapshot dataSnapshot) {
```

```
                            Geocoder geocoder;
                            List<Address> addresses;
                            geocoder = new
Geocoder(getContext(), Locale.getDefault());
                            try {
                                addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                                _address =
addresses.get(0).getAddressLine(0);
                            } catch (IOException e)
{
                                e.printStackTrace();
                            }

                            ModelFleet fleet =
dataSnapshot.getValue(ModelFleet.class);
                            String address =
_address;
                            String customer =
task.getCustomer();
                            String driverName =
fleet.getFullname();
                            String departureID =
task.getDepartureID();
                            String workType =
task.getWorkType();
                            OngoinglistMap.put(key,
new ModelOngoingList(customer, address, departureID,
driverName, workType));

mAdapter.notifyDataSetChanged();
                            checkEmpty();
                        }

                        @Override
                        public void
onCancelled(DatabaseError databaseError) {

                        }
                    };
                    new
FireDataUserFleet().ref.child(task.getFleetID()).add
ListenerForSingleValueEvent(singleEventListener);
                }
                else{
                    final String key =
dataSnapshot.getKey();
```

```
                    OngoinglistMap.remove(key);
                    mAdapter.notifyDataSetChanged();
                    checkEmpty();
                }
            }
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            String key = dataSnapshot.getKey();
            OngoinglistMap.remove(key);
            mAdapter.notifyDataSetChanged();
            checkEmpty();
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.addChildEventListener(childEventL
istener);
}
```

Gambar 5.58 US-AM2-4: Mengambil data Task

Data tiap tugas dimasukkan ke dalam *HashMap* ongoing*List*Map. *HashMap* ini nantinya dimasukkan ke dalam *adapter* yang memungkinkan 1 key pada *HashMap* ditampilkan dalam 1 *card*. *Adapter* yang akan ditampilkan dalam *RecyclerView recyclerView_ongoing*. Apabila tidak ada tugas yang terdata, *RecyclerView* tidak muncul dan sebagai gantinya memunculkan *TextView* yang menjelaskan tidak ada penugasan yang terekam. Kode yang digunakan adalah pada gambar 5.59. Konfirgurasi pada *AdapterView* adalah pada gambar 5.60.

Adapter tersebut menggunakan *card_task_ongoing* dengan tampilan pada gambar 5.61.

```
private void initRecycler(){
    mAdapter = new
TaskOngoingAdapter(OngoinglistMap);
    Log.d("tsss", "task on Going cycler: " +
OngoinglistMap.toString());
    RecyclerView.LayoutManager mLayoutManager = new
LinearLayoutManager(getActivity());

recyclerView_ongoing.setLayoutManager(mLayoutManager
);
    recyclerView_ongoing.setItemAnimator(new
DefaultItemAnimator());
    recyclerView_ongoing.setAdapter(mAdapter);

    checkEmpty();

}

private void checkEmpty(){
    if(OngoinglistMap.size()>0){

recyclerView_ongoing.setVisibility(View.VISIBLE);

tv_emptyOngoingTask.setVisibility(View.GONE);
    }
    else {

recyclerView_ongoing.setVisibility(View.GONE);

tv_emptyOngoingTask.setVisibility(View.VISIBLE);
    }
}
```

Gambar 5.59 US-AM2-4: set Adapter pada recycler view

```
private Map<String, ModelOngoingList> taskOngoingMap
= new LinkedHashMap<>();
private OnItemClickListener mOnItemClickListener;

public interface OnItemClickListener {
    void onItemClick(View view, ModelFleet obj, int
position, String key);
}

public void setOnItemClickListener(final
OnItemClickListener mItemClickListener) {
    this.mOnItemClickListener = mItemClickListener;
}


public class MyViewHolder extends
RecyclerView.ViewHolder{

    @BindView(R.id.tv_customername) TextView
tv_customername;
    @BindView(R.id.tv_address) TextView tv_address;
    @BindView(R.id.tv_departureID) TextView
tv_departureID;
    @BindView(R.id.tv_notfinished_type) TextView
tv_notfinished_type;
    @BindView(R.id.rlCard) LinearLayout rlCard;

    public MyViewHolder(View view) {
        super(view);
        ButterKnife.bind(this, view);
    }
}

public TaskOngoingAdapter(Map<String,
ModelOngoingList> taskOngoingMap){
    this.taskOngoingMap = taskOngoingMap;
}

@Override
public MyViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
    View itemview =
LayoutInflater.from(parent.getContext()).
            inflate(R.layout.card_task_ongoing,
parent, false);
    return new MyViewHolder(itemview);
}
```

```
@Override
public void onBindViewHolder(MyViewHolder holder,
int position) {
    final List<String> keyList = new
ArrayList<>(taskOngoingMap.keySet());
    final List<ModelOngoingList> taskOngoingList =
new ArrayList<>(taskOngoingMap.values());
    final String key = keyList.get(position);
    ModelOngoingList taskOngoing =
taskOngoingList.get(position);
    String departureID;
    if(taskOngoing.getDepartureID()==null){
        departureID = "-";
    }
    else{
        departureID = taskOngoing.getDepartureID();
    }


holder.tv_customername.setText(taskOngoing.getCustom
er());

holder.tv_address.setText(taskOngoing.getAddress());
    holder.tv_departureID.setText("Departure ID: " +
departureID);

holder.tv_notfinished_type.setText(taskOngoing.getWo
rkType());

    holder.rlCard.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new
Intent(v.getContext().getApplicationContext(),
OnGoingDetail.class);
            intent.putExtra("taskID", key);
            v.getContext().startActivity(intent);
        }
    });
}

@Override
public int getItemCount() {
    return taskOngoingMap.size();
}
```

Gambar 5.60 US-AM2-4: TaskOngoingAdapter

Gambar 5.61 US-AM2-4: card_task_ongoing

Saat salah satu *card* diklik, maka masuk ke *activity* detail *task*. Aktivitas ini memiliki 2 *fragment*, yakni deskripsi penugasan bernama Ongoing_Description dan riwayat rute yang dilalui bernama Ongoing_Route. Untuk memuat produk digunakan kode pada gambar 5.63.

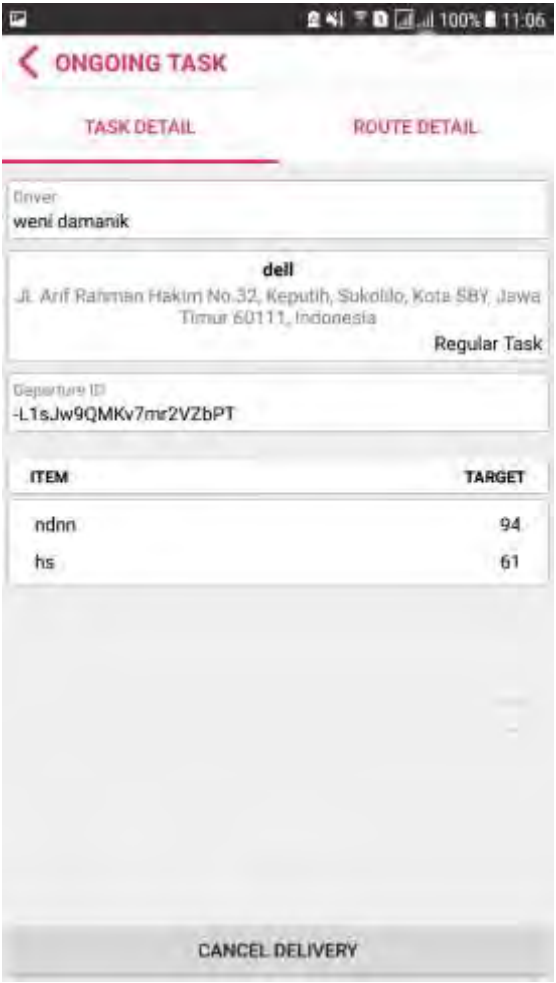
```
private void loadproduct(){
    singleEventListener1 = new ValueEventListener()
{
        @Override
        public void onDataChange(final DataSnapshot
dataSnapshot) {
            final ModelTask task =
dataSnapshot.getValue(ModelTask.class);
            customer = task.getCustomer();
            final ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.
class);

            Geocoder geocoder;
            List<Address> addresses;
            geocoder = new Geocoder(getContext(),
Locale.getDefault());
            try {
                addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                address =
addresses.get(0).getAddressLine(0);
            } catch (IOException e) {
                e.printStackTrace();
            }

            singleEventListener2 = new
ValueEventListener() {
                @Override
                public void
onDataChange(DataSnapshot dataSnapshot1) {
                    final ModelFleet fleet =
dataSnapshot1.getValue(ModelFleet.class);
                    fleetID=dataSnapshot1.getKey();

                    valueEventListener = new
ValueEventListener() {
                        @Override
                        public void
onDataChange(DataSnapshot dataSnapshot2) {
                            String departure;

if(task.getDepartureID()==null){
                                departure = "-";
                            }
                            else{
                                departure =
```

```
task.getDepartureID();
                                departureID =
task.getDepartureID();
                            }


tv_ongoing_drivername.setText(fleet.getFullname());

tv_ongoing_customername.setText(task.getCustomer());

tv_ongoing_address.setText(address);

tv_ongoing_departureID.setText(departure);

tv_taskType.setText(task.getWorkType());

                            for(DataSnapshot
postSnapshot :  dataSnapshot2.getChildren()){
                                ModelMessage
modelmessage =
postSnapshot.getValue(ModelMessage.class);
                                LayoutInflater
_canvasProduct = LayoutInflater.from(getContext());
                                View canvasProduct =
_canvasProduct.inflate(R.layout.card_item_ongoing,
null, false);
                                TextView
tv_ongoing_productname = (TextView)
canvasProduct.findViewById(R.id.tv_ongoing_productna
me);
                                TextView
tv_ongoing_productamount = (TextView)
canvasProduct.findViewById(R.id.tv_ongoing_productam
ount);


tv_ongoing_productname.setText(modelmessage.getItem(
));

tv_ongoing_productamount.setText(modelmessage.getTar
get());


ll_ongoing_product.addView(canvasProduct);

ll_ongoing_product.requestLayout();
                            }
                        }
```

```
                        @Override
                        public void
onCancelled(DatabaseError databaseError) {

                        }
                    };
                    new
FireDataTask().ref.child(taskID).child("message").ad
dValueEventListener(valueEventListener);

                }

                @Override
                public void
onCancelled(DatabaseError databaseError) {

                }
            };
            new
FireDataUserFleet().ref.child(task.getFleetID()).add
ListenerForSingleValueEvent(singleEventListener2);
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSingl
eValueEvent(singleEventListener1);
}
```

Gambar 5.62 US-AM2-4: Memuat detail penugasan

User interface untuk fragment yang menampilkan detail deskripsi penugasan adalah pada gambar 5.63.



Gambar 5.63 US-AM2-4: Tampilan detail deskripsi penugasan

Kemudian untuk memuat kordinat-kordinat pada *Route_Detail*, digunakan kode pada gambar 5.64.

```
private void getLocation(){
    taskEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            ModelTask task =
dataSnapshot.getValue(ModelTask.class);
            ModelAddress modelAddress =
dataSnapshot.child(FireDataTask.ADDRESS).getValue(Model
Address.class);
            ModelAddress modelWareHouse =
dataSnapshot.child(FireDataTask.WAREHOUSELOC).getValue(
ModelAddress.class);
            latLngWareHouse = new
LatLng(modelWareHouse.getLatitude(),
modelWareHouse.getLongitude());
            latLngCustomer = new
LatLng(modelAddress.getLatitude(),
modelAddress.getLongitude());

            mMap.clear();
            customerMarker = mMap.addMarker(new
MarkerOptions().position(latLngCustomer).title(task.get
Customer()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_customer", 52, 100))));
            wareHouseMarker = mMap.addMarker(new
MarkerOptions().position(latLngWareHouse).title(task.ge
tWareHouseName()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_warehouse", 52, 100))));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLn
gCustomer, 12));


if(dataSnapshot.child(FireDataTask.LOCATION).getValue()
!= null){

                trackEventListener = new
ChildEventListener() {
                    @Override
                    public void
onChildAdded(DataSnapshot dataSnapshot, String s) {
                        ModelAddress _track =
dataSnapshot.getValue(ModelAddress.class);
                        LatLng track = new
LatLng(_track.getLatitude(), _track.getLongitude());
                        mMap.addMarker(new
MarkerOptions().position(track).
```

```
icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_fleet_old_location", 30, 30))));
                    }

                    @Override
                    public void
onChildChanged(DataSnapshot dataSnapshot, String s) {
                        ModelAddress _track =
dataSnapshot.getValue(ModelAddress.class);
                        LatLng track = new
LatLng(_track.getLatitude(), _track.getLongitude());
                        mMap.addMarker(new
MarkerOptions().position(track).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_fleet_old_location", 30, 30))));
                    }

                    @Override
                    public void
onChildRemoved(DataSnapshot dataSnapshot) {
}

                    @Override
                    public void
onChildMoved(DataSnapshot dataSnapshot, String s) {
}

                    @Override
                    public void
onCancelled(DatabaseError databaseError) {
}
                };
                new
FireDataTask().ref.child(taskID).child(FireDataTask.LOC
ATION).addChildEventListener(trackEventListener);
            }

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {
        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSingleVa
lueEvent(taskEventListener);
}
```

Gambar 5.64 US-AM2-4: Memuat detail kordinat lokasi

Untuk *marker* lintasan yang dilalui armada, digunakan *ChildEventListener* karena nilainya selalu bertambah saat sedang diantarkan. User interface untuk fragment yang menampilkan detail rute adalah pada gambar 5.65.

Gambar 5.65 US-AM2-4: Tampilan detail kordinat lokasi

5.2.2.10 US-AM2-5: Melihat riwayat penugasan yang sudah berakhir

Tugas-tugas yang sudah diselesaikan ditampilkan dalam *fragment Finished_list* yang disajikan dalam barisan *card*. Tiap *card* memiliki konten nama pelanggan, *departureID*, alamat pelanggan, tipe penugasan (*regular task* dan *rework task*), dan kategori keberhasilan (*successfully delivered*, *unsuccessfully delivered*, *rework with same Fleet*, *rework with other Fleet*). Semua data tersedia dalam 1 ranting *Task* di *Firebase* jadi hanya perlu mengambil 1 data saja. Data yang diambil adalah yang bernilai **selain *false*** di ranting *Task→TaskID→isCompleted*. Pengambilan data *Task* menggunakan fungsi pada gambar 5.66.

Data tugas tersebut dimasukkan ke dalam *HashMap*. Semua tugas harus dikirimkan dengan jumlah yang sesuai dengan saat penugasan diberikan. Oleh karena itu ada 2 *HashMap* yang pada *list* ini yakni untuk menampung semua tugas yang sudah diantarkan dan tugas yang belum berhasil diselesaikan. Semua *task* yang sudah diantarkan memiliki nilai **selain *false*** lalu dimasukkan pada *task*FinishedMap1 dan yang belum berhasil bernilai ***some item not delivered*** lalu dimasukkan pada *taskFinishedMap2*.*taskFinishedMap1* dimasukkan ke dalam *TaskFinishedAdapter* dan dinamakan *mAdapter1*. Kemudian *mAdapter1* ditampilkan di *RecyclerView recyclerview_finished*. *taskFinishedMap2* dimasukkan ke dalam *TaskFinishedAdapter* dan dinamakan *mAdapter2*. Kemudian *mAdapter2* ditampilkan di *RecyclerView recyclerview_finished2*. Kode yang digunakan adalah pada gambar 5.67. Kemudian konfigurasi pada *AdapterView* adalah pada gambar 5.68.

```
private void initTaskList(){
    childEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {

if(adminKey.equals(dataSnapshot.child("companyID").g
etValue().toString())){

if(!dataSnapshot.child("isCompleted").getValue().toS
tring().equals("false")){
                    String key  =
dataSnapshot.getKey();
                    ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.
class);

                    String _address=null;
                    Geocoder geocoder;
                    List<Address> addresses;
                    geocoder = new
Geocoder(getContext(), Locale.getDefault());
                    try {
                        addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                        _address =
addresses.get(0).getAddressLine(0);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    String customer =
task.getCustomer();
                    String departure =
task.getDepartureID();
                    String address = _address;
                    String status =
task.getIsCompleted();
                    String type =
task.getWorkType();
                    taskFinishedMap1.put(key, new
ModelFinishedList(customer, departure, address,
status, type));

mAdapter1.notifyDataSetChanged();
```

```
                    checkEmpty();
                }

if(dataSnapshot.child("isCompleted").getValue().toSt
ring().equals("unsuccessfully delivered")){
                    String key  =
dataSnapshot.getKey();
                    ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.
class);

                    String _address=null;
                    Geocoder geocoder;
                    List<Address> addresses;
                    geocoder = new
Geocoder(getContext(), Locale.getDefault());
                    try {
                        addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                        _address =
addresses.get(0).getAddressLine(0);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    String customer =
task.getCustomer();
                    String departure =
task.getDepartureID();
                    String address = _address;
                    String status =
task.getIsCompleted();
                    String type =
task.getWorkType();
                    Log.d("tsss", "Finished task:
"+key);
                    taskFinishedMap2.put(key, new
ModelFinishedList(customer, departure, address,
status, type));

mAdapter2.notifyDataSetChanged();
                    checkEmpty();
                }
            }
        }
```

```
        @Override
        public void onChildChanged(DataSnapshot
dataSnapshot, String s) {

if(adminKey.equals(dataSnapshot.child("companyID").g
etValue().toString())){

if(!dataSnapshot.child("isCompleted").getValue().toS
tring().equals("false")){
                    String key  =
dataSnapshot.getKey();
                    ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.
class);

                    String _address=null;
                    Geocoder geocoder;
                    List<Address> addresses;
                    geocoder = new
Geocoder(getContext(), Locale.getDefault());
                    try {
                        addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                        _address =
addresses.get(0).getAddressLine(0);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    String customer =
task.getCustomer();
                    String departure =
task.getDepartureID();
                    String address = _address;
                    String status =
task.getIsCompleted();
                    String type =
task.getWorkType();
                    Log.d("tsss", "Finished task:
"+key);
                    taskFinishedMap1.put(key, new
ModelFinishedList(customer, departure, address,
status, type));
```

```
mAdapter1.notifyDataSetChanged();
                    checkEmpty();
                }
                else
if(!dataSnapshot.child("isCompleted").getValue().toS
tring().equals("some item not delivered")){
                    String key  =
dataSnapshot.getKey();
                    ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.
class);

                    String _address=null;
                    Geocoder geocoder;
                    List<Address> addresses;
                    geocoder = new
Geocoder(getContext(), Locale.getDefault());
                    try {
                        addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                        _address =
addresses.get(0).getAddressLine(0);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    String customer =
task.getCustomer();
                    String departure =
task.getDepartureID();
                    String address = _address;
                    String status =
task.getIsCompleted();
                    String type =
task.getWorkType();
                    taskFinishedMap2.put(key, new
ModelFinishedList(customer, departure, address,
status, type));

mAdapter2.notifyDataSetChanged();
                    checkEmpty();
                }
                else{
                    final String key =
dataSnapshot.getKey();
```

```
                    taskFinishedMap1.remove(key);
                    taskFinishedMap2.remove(key);

mAdapter1.notifyDataSetChanged();

mAdapter2.notifyDataSetChanged();
                    checkEmpty();
                }
            }
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            String key = dataSnapshot.getKey();
            taskFinishedMap1.remove(key);
            taskFinishedMap2.remove(key);
            mAdapter1.notifyDataSetChanged();
            mAdapter2.notifyDataSetChanged();
            checkEmpty();
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.addChildEventListener(childEventL
istener);
}
```

Gambar 5.66 US-AM2-5: Mengambil data task

```
rivate void initRecycler(){
    mAdapter1 = new
TaskFinishedAdapter(taskFinishedMap1);
    RecyclerView.LayoutManager mLayoutManager1 = new
LinearLayoutManager(getActivity());

recyclerview_finished.setLayoutManager(mLayoutManager1)
;
    recyclerview_finished.setItemAnimator(new
DefaultItemAnimator());
    recyclerview_finished.setAdapter(mAdapter1);

    mAdapter2 = new
TaskFinishedAdapter(taskFinishedMap2);
    RecyclerView.LayoutManager mLayoutManager2 = new
LinearLayoutManager(getActivity());

recyclerview_finished2.setLayoutManager(mLayoutManager2
);
    recyclerview_finished2.setItemAnimator(new
DefaultItemAnimator());
    recyclerview_finished2.setAdapter(mAdapter2);

    checkEmpty();
}

private void checkEmpty(){
    if(taskFinishedMap1.size()>0){

recyclerview_finished.setVisibility(View.VISIBLE);

recyclerview_finished2.setVisibility(View.GONE);
        tv_emptyFinishedTask.setVisibility(View.GONE);
    }
    else {
        recyclerview_finished.setVisibility(View.GONE);

recyclerview_finished2.setVisibility(View.GONE);

tv_emptyFinishedTask.setVisibility(View.VISIBLE);
    }
}
```

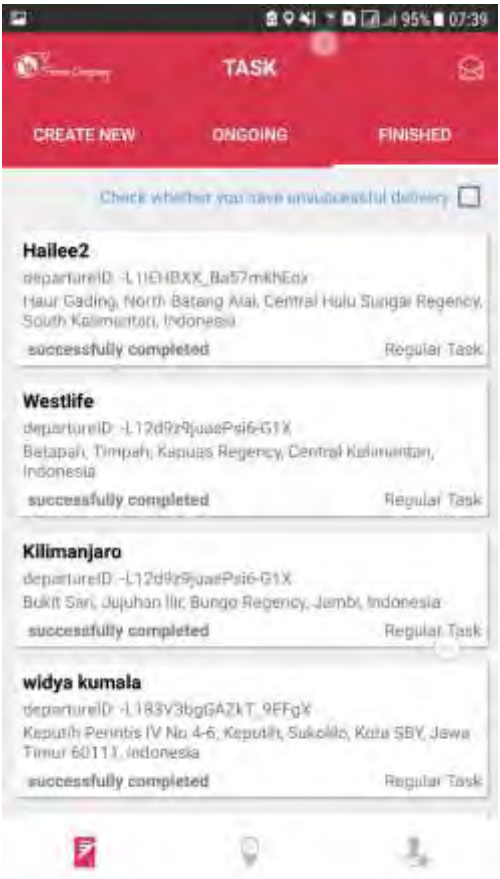
Gambar 5.67 US-AM2-5: Set adapter pada recyler view

```
private Map<String, ModelFinishedList> taskFinishedMap
= new LinkedHashMap<>();
private FleetAccountAdapter.OnItemClickListener
mOnItemClickListener;

public interface OnItemClickListener {
    void onItemClick(View view, ModelFleet obj, int
position, String key);
}

public void setOnItemClickListener(final
FleetAccountAdapter.OnItemClickListener
mItemClickListener) {
    this.mOnItemClickListener = mItemClickListener;
}

public class MyViewHolder extends
RecyclerView.ViewHolder{

    @BindView(R.id.tv_customer) TextView tv_customer;
    @BindView(R.id.tv_departureID) TextView
tv_departureID;
    @BindView(R.id.tv_address) TextView tv_address;
    @BindView(R.id.rlCard) RelativeLayout rlCard;
    @BindView(R.id.tv_finished_status) TextView
tv_finished_status;
    @BindView(R.id.tv_finished_type) TextView
tv_finished_type;

    public MyViewHolder(View itemView) {
        super(itemView);
        ButterKnife.bind(this, itemView);
    }
}

public TaskFinishedAdapter(Map<String,
ModelFinishedList> taskFinishedMap){
    this.taskFinishedMap = taskFinishedMap;
}

@Override
public MyViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
    View itemview =
LayoutInflater.from(parent.getContext()).

inflate(R.layout.card_task_finished,parent,false);

    return new MyViewHolder(itemview);
}

@Override
```

```
public void onBindViewHolder(MyViewHolder holder, int
position) {
    final List<String> keyList = new
ArrayList<>(taskFinishedMap.keySet());
    final List<ModelFinishedList> finishedTaskList =
new ArrayList<>(taskFinishedMap.values());
    final String key = keyList.get(position);
    ModelFinishedList taskFinished=
finishedTaskList.get(position);
    String departureID;
    if(taskFinished.getDeparture()==null){
        departureID = "-";
    }
    else{
        departureID = taskFinished.getDeparture();
    }


holder.tv_customer.setText(taskFinished.getCustomer());

holder.tv_address.setText(taskFinished.getAddress());
    holder.tv_departureID.setText("departureID: " +
departureID);

holder.tv_finished_status.setText(taskFinished.getStatu
s());

holder.tv_finished_type.setText(taskFinished.getType())
;

    holder.rlCard.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Log.d("tssss", "key extra: "+key );
            Intent intent = new
Intent(v.getContext().getApplicationContext(),
Finished_Detail.class);
            intent.putExtra("taskID", key);
            v.getContext().startActivity(intent);
        }
    });
}


@Override
public int getItemCount() {
    return taskFinishedMap.size();
}
```

Gambar 5.68 US-AM2-5: TaskFinishedAdapter

Adapter tersebut menggunakan card_task_finished dengan tampilan pada gambar 5.69.



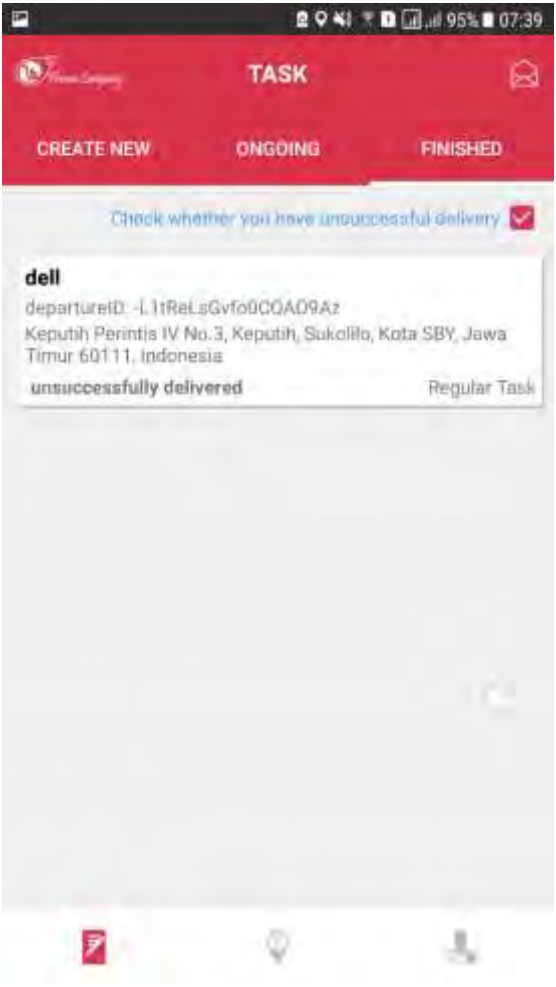
Gambar 5.69 US-AM2-5: card_task_finished

Secara *default recyclerview_finished* saja yang ditampilkan, sedangkan *recyclerview_finished2* dihilangkan. Untuk memunculkannya, maka harus menyentang *CheckBox* pada sudut atas *fragment* menggunakan kode pada gambar 5.70.

```
@Override public void onStart(){
    super.onStart();
    checkbox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void
onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
            if(isChecked){
                if(taskFinishedMap2.size()>0){

recyclerview_finished.setVisibility(View.GONE);

recyclerview_finished2.setVisibility(View.VISIBLE
);

tv_emptyFinishedTask.setVisibility(View.GONE);
                }
                 else {

recyclerview_finished.setVisibility(View.GONE);

recyclerview_finished2.setVisibility(View.GONE);

tv_emptyFinishedTask.setVisibility(View.VISIBLE);
                }
                Log.d("tsss", "Finished list
checked");
            }
            else{
                Log.d("tsss", "Finished list
unchecked");
                checkEmpty();
            }
        }
    });
}
```

Gambar 5.70 US-AM2-5: Menampilkan recyclerview_finished2

Saat dicentang, *recyclerview_finished2* akan ditampilkan pada gambar 5.71. Pada gambar tersebut,



Gambar 5.71 US-AM2-5: card_task_finished

Saat salah satu *card* diklik, maka masuk ke *activity* detail *task*. Aktivitas ini memiliki 2 *fragment*, yakni deskripsi penugasan bernama *Finished_Description* dan riwayat rute yang dilalui

bernama *Finished_Route*. Untuk memuat produk digunakan kode pada gambar 5.72.

```
private void loadProduct() {
    singleEventListener1 = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot1) {
            final ModelTask task=
dataSnapshot1.getValue(ModelTask.class);
            final ModelAddress custAddress =
dataSnapshot1.child(FireDataTask.ADDRESS).getValue(Mode
lAddress.class);
            final ModelAddress wareHouseAddress =
dataSnapshot1.child(FireDataTask.WAREHOUSELOC).getValue
(ModelAddress.class);
             singleEventListener2 = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot2) {
                    final ModelFleet fleet =
dataSnapshot2.getValue(ModelFleet.class);
                    Geocoder geocoder;
                    List<Address> addresses;
                    geocoder = new
Geocoder(getContext(), Locale.getDefault());
                    try {
                        addresses =
geocoder.getFromLocation(custAddress.getLatitude(),
custAddress.getLongitude(),1);
                        address =
addresses.get(0).getAddressLine(0);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    latLngAddress = new
LatLng(custAddress.getLatitude(),
custAddress.getLongitude());
                    companyID = task.getCompanyID();
                    customer = task.getCustomer();
                    fleetID = task.getFleetID();
                    fleetName = fleet.getFullname();
                    wareHouseLoc = new
LatLng(wareHouseAddress.getLatitude(),
wareHouseAddress.getLongitude());
                    wareHouseName =
task.getWareHouseName();
```

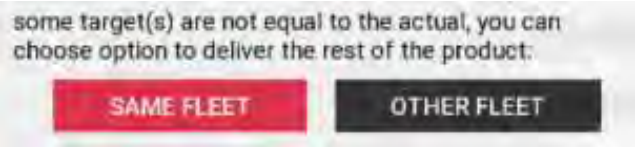
```
tv_finished_address.setText(address);

tv_finished_customername.setText(customer);

tv_finished_departureID.setText(task.getDepartureID());

tv_finished_drivername.setText(fleetName);

tv_finished_status.setText(task.getIsCompleted());

tv_finished_type.setText(task.getWorkType());
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataUserFleet().ref.child(task.getFleetID()).addLis
tenerForSingleValueEvent(singleEventListener2);

            valueEventListener = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot4) {
                    for(DataSnapshot postSnapshot :
dataSnapshot4.getChildren()){
                        ModelMessage modelMessage =
postSnapshot.getValue(ModelMessage.class);
                        Log.d("tsss", "product load: "
+ postSnapshot.getKey()+ " " + modelMessage.getItem()+
modelMessage.getTarget());
                        LayoutInflater _canvasProduct =
LayoutInflater.from(getContext());
                        View canvasProduct =
_canvasProduct.inflate(R.layout.card_item_finished,
null, false);
                        TextView
tv_finished_productname = (TextView)
canvasProduct.findViewById(R.id.tv_finished_productname
);
                        TextView
tv_finished_producttarget = (TextView)
canvasProduct.findViewById(R.id.tv_finished_producttarg
et);
                        TextView
tv_finished_productactual = (TextView)
canvasProduct.findViewById(R.id.tv_finished_productactu
```

```
al);


tv_finished_productname.setText(modelMessage.getItem())
;

tv_finished_producttarget.setText(modelMessage.getTarge
t());

tv_finished_productactual.setText(modelMessage.getActua
l());


ll_finished_product.addView(canvasProduct);

ll_finished_product.requestLayout();


if(task.getIsCompleted().equals("unsuccessfully
delivered")){

if(Integer.parseInt(modelMessage.getActual())<Integer.p
arseInt(modelMessage.getTarget())){
                                String newItem =
modelMessage.getItem();
                                int newTarget =
Integer.parseInt(modelMessage.getTarget())-
Integer.parseInt(modelMessage.getActual());
                                listSize++;
                                Map<String, Object>
objectMessage = new HashMap<>();

objectMessage.put(FireDataTask.MESSAGE_VALUE_ITEM,
newItem);

objectMessage.put(FireDataTask.MESSAGE_VALUE_TARGET,
Integer.toString(newTarget));

messageMap.put(Integer.toString(listSize),
objectMessage);

                            }

llRework.setVisibility(View.VISIBLE);
                        }
                        else{

llRework.setVisibility(View.GONE);
                        }
                    }
                }
```

```
                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataTask().ref.child(taskID).child("message").addVa
lueEventListener(valueEventListener);
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSingleVa
lueEvent(singleEventListener1);
}
```

Gambar 5.72 US-AM2-5: Memuat detail penugasan

Aplikasi akan mengecek apabila ranting *isCompleted* bernilai "*unsuccessfully delivered*", maka muncul *Layout* yang memberikan opsi untuk pengerjaan ulang dengan nama llRework. Ada *HashMap messageMap* untuk menampung item-item yang nilai dari apabila barang yang sampai lebih sedikit daripada seharusnya. *messageMap* ini digunakan untuk membuat penugasan baru dengan kategori *reworkTask*. *llRework* memiliki tampilan seperti gambar 5.73.



Gambar 5.73 US-AM2-5: llRework

Fragment Finished_Description memiliki tampilan seperti gambar 5.74, dimana perbedaannya dengan Fragment

Not_Finished_Description adalah ada nilai actual yakni jumlahh barang yang sampai pada pelanggan.

Gambar 5.74 US-AM2-5: Tampilan detail penugasan

Kemudian untuk memuat kordinat-kordinat pada *Route_Detail*, digunakan kode pada gambar 5.75. Fragment tersebut memiliki tampilan seperti gambar 5.76.

```
private void getDatabase(){
    taskEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            ModelTask task =
dataSnapshot.getValue(ModelTask.class);
            ModelAddress modelAddress =
dataSnapshot.child(FireDataTask.ADDRESS).getValue(Mo
delAddress.class);
            ModelAddress modelWareHouse =
dataSnapshot.child(FireDataTask.WAREHOUSELOC).getVal
ue(ModelAddress.class);
            latLngCustomer = new
LatLng(modelAddress.getLatitude(),
modelAddress.getLongitude());
            latLngWareHouse = new
LatLng(modelWareHouse.getLatitude(),
modelWareHouse.getLongitude());

            mMap.clear();
            customerMarker = mMap.addMarker(new
MarkerOptions().position(latLngCustomer).title(task.
getCustomer()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap
("map_customer", 52, 100))));
            wareHouseMarker = mMap.addMarker(new
MarkerOptions().position(latLngWareHouse).title(task
.getWareHouseName()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap
("map_warehouse", 52, 100))));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(la
tLngCustomer, 12));



if(dataSnapshot.child(FireDataTask.LOCATION).getValu
e() != null){
```

```
                trackEventListener = new
ValueEventListener() {
                    @Override
                    public void
onDataChange(DataSnapshot dataSnapshot) {
                        for(DataSnapshot
postSnapshot : dataSnapshot.getChildren()){
                            ModelAddress _track =
postSnapshot.getValue(ModelAddress.class);
                            LatLng track = new
LatLng(_track.getLatitude(), _track.getLongitude());
                            mMap.addMarker(new
MarkerOptions().position(track).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap
("map_fleet_old_location", 30, 30))));
                        }
                    }

                    @Override
                    public void
onCancelled(DatabaseError databaseError) {

                    }
                };
                new
FireDataTask().ref.child(taskID).child(FireDataTask.
LOCATION).addValueEventListener(trackEventListener);
            }
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSingl
eValueEvent(taskEventListener);
}
```

Gambar 5.75 US-AM2-5: Memuat detail kordinat lokasi

Gambar 5.76 US-AM2-5: Tampilan detail kordinat lokasi

5.2.2.11 US-AM3-1: Menerima laporan barang sampai tanpa manipulasi

Satu-satunya keamanan yang dipakai oleh aplikasi agar penyelesaian tugas dikonfirmasi oleh pelanggan langsung, bukan sopir atau orang lain adalah dengan menggunakan token. Token yang dibuat ketika membuat *task* baru. Setelah tugas

selesai dibuat, *user* harus memberitahu pelanggan Token yang harus mereka isikan saat barang sampai.

5.2.2.12 US-AM4-1: Membuat penugasan lanjutan pada sopir yang sama

Pada gambar 5.73, saat tombol **same *Fleet*** ditekan, perintah yang dijalankan adalah pada gambar 5.77.

```
@OnClick(R.id.btn_sameFleet) public void
reworkSameFleet() {
    new
FireDataTempTask().writeTempTaskSameFleet(wareHouseName
, wareHouseLoc, companyID, fleetID, latLngAddress,
customer, messageMap,
            new DatabaseReference.CompletionListener()
{
        @Override
        public void onComplete(DatabaseError
databaseError, final DatabaseReference
databaseReference1) {
            Intent intent = new Intent(getContext(),
Rework_Task.class);
            intent.putExtra("tempTaskID",
databaseReference1.getKey());
            intent.putExtra("fleetID", fleetID);
            intent.putExtra("oldTaskID", taskID);
            intent.putExtra("fleetName", fleetName);
            startActivity(intent);
        }
    });
}
```

Gambar 5.77 US-AM4-1: Perintah penugasan lanjutan pada sopir yang sama

Perintah tersebut akan menyimpan data penugasan sementara pada *Firebase* dengan ranting *temp_task* yang desainnya sama persis dengan *task*. Setelah itu masuk ke *activity Rework_Task*. Karena *intent extra* saat membuka aplikasi ini memuat *FleetID* dan FleetName, *Rework_Task* tidak memunculkan daftar armada dalam *spinner* namun *TextView* yang menampilkan nama sopir sehingga sopir tidak diganti-ganti lagi. Tampilan *activity* ini sama dengan pembuatan tugas baru. Nama

pelanggan, alamat, dan daftar barang sudah terisi. *User* hanya perlu mengisikan token, dan gudang. Setelah diisi, barulah bisa membuat penugasan baru dengan kode pada gambar 5.79.

```
@OnClick(R.id.btn_createTask) public void createTask(){
    token = et_edit_token.getText().toString();
    customer = et_edit_customer.getText().toString();
    if(TextUtils.isEmpty(token)){
        Toast.makeText(this, "Please enter Full Name",
Toast.LENGTH_SHORT).show();
        return;
    }
    if(TextUtils.isEmpty(customer)){
        Toast.makeText(this, "Please enter Full Name",
Toast.LENGTH_SHORT).show();
        return;
    }

    for(int i=0; i<newMessageList.size(); i++){
        String id = newMessageList.get(i).getId() + "";
        String item =
newMessageList.get(i).getEtItem().getText() + "";
        String amount =
newMessageList.get(i).getEtTarget().getText() + "";

        Map<String, Object> objectMessage = new
HashMap<>();

objectMessage.put(FireDataTask.MESSAGE_VALUE_ITEM,
item);

objectMessage.put(FireDataTask.MESSAGE_VALUE_TARGET,
amount);

        messageMap.put(id, objectMessage);
        if(i == newMessageList.size()-1){
            new
FireDataTask().writeReworkTask(companyID,
wareHouseName, latLngWareHouse, fleetID, latLngAddress,
customer, token, messageMap, new
DatabaseReference.CompletionListener() {
                @Override
                public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                    if(tasktype.equals("same fleet")){
                        new
FireDataTask().ReworkTaskSameFleet(oldTaskID, new
DatabaseReference.CompletionListener() {
                            @Override
                            public void
```

```
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference2) {
                                    new
FireDataTempTask().ref.child(taskID).removeValue();
                                    startActivity(new
Intent(getApplicationContext(), HomeActivity.class));
                                    finish();
                                }
                            });
                        }
                        else if(tasktype.equals("other
fleet")){
                            new
FireDataTask().ReworkTaskAnotherFleet(oldTaskID, new
DatabaseReference.CompletionListener() {
                                @Override
                                public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference2) {
                                    new
FireDataTempTask().ref.child(taskID).removeValue();
                                    startActivity(new
Intent(getApplicationContext(), HomeActivity.class));
                                    finish();
                                }
                            });
                        }
                    }
                });
            }
        }
}
```

Gambar 5.78 US-AM4-1: Menyimpan penugasan baru di Firebase

5.2.2.13 US-AM4-2: Membuat penugasan lanjutan pada sopir yang berbeda

Pada gambar 5.73, saat tombol **other *Fleet*** ditekan, perintah yang dijalankan adalah pada gambar 5.79.

```
@OnClick(R.id.btn_otherFleet) public void
reworkOtherFleet(){
    new
FireDataTempTask().writeTempTaskOtherFleet(wareHo
useName, wareHouseLoc, companyID, latLngAddress,
customer, messageMap,
            new
DatabaseReference.CompletionListener() {
                @Override
                public void
onComplete(DatabaseError databaseError, final
DatabaseReference databaseReference1) {
                    Intent intent = new
Intent(getContext(), Rework_Task.class);
                    intent.putExtra("tempTaskID",
databaseReference1.getKey());
                    intent.putExtra("oldTaskID",
taskID);
                    startActivity(intent);
                }
            });
}
```

Gambar 5.79 US-AM4-2: Perintah penugasan lanjutan pada sopir berbeda

Perintah tersebut akan menyimpan data penugasan sementara pada *Firebase* dengan ranting temp_*task* yang desainnya sama persis dengan *task*. Setelah itu masuk ke *activity* Rework_*Task*. Karena *intent extra* saat membuka aplikasi ini tidak memuat *FleetID* dan *FleetName*, *Rework_Task* memunculkan daftar armada dalam *spinner* seperti *activity New_Create*. Nama pelanggan, alamat, dan daftar barang yang harus diantarkan sudah terisi. *User* hanya perlu mengisikan sopir, token, dan gudang. Setelah diisi, barulah bisa membuat penugasan baru dengan kode pada gambar 5.78.

5.2.2.14 US-AM4-3: Mengalihkan penugasan ke sopir lain

Saat berhalangan mengantarkan barang, sopir harus mengajukan pengalihan penugasan ke sopir lain. *User* dapat menyetujui dan mencarikan sopir lain atau malah menolaknya.

Untuk melihat adanya pengajuan dari sopir, *user* melihat *list* pengajuan dalam bentuk pesan pada *activity MessageActivity*. Pesan disajikan dalam barisan *card*. Tiap *card* memiliki konten yakni sopir yang mengajukan dan jenis pengajuan (*Task* cancellation, *departure* cancellation). Setelah diklik maka masuk ke *activity Message*Detail. Aplikasi mengambil data pesan dari ranting *Firebase* "*message*".

Apabila ranting *Message*→*companyID*→*FleetID*→*message*ID →*taskID* memiliki nilai, maka Layout detail *task* ditampilkan sedangkan Layout detail *departure* dihilangkan.

Sebaliknya apabila ranting *Message*→*companyID*→*FleetID*→ *message*ID→*DepartureID* yang memiliki nilai, maka Layout detail *departure* yang ditampilkan.

Kemudian apabila ranting *Message*→*companyID*→*FleetID*→ *message*ID→Response tidak memiliki nilai, maka muncul tombol untuk menyetujui atau menolak pengajuan seperti pada gambar 5.81. Sedangkan jika memiliki nilai, maka tidak ditampilkan agar tugas yang sudah dialihkan tidak bisa dialihkan kembali. Jika sudah ada respon, maka aplikasi menampilkan apakah pengajuan tersebut ditolak atau disetujui seperti pada gambar 5.84. Jika disetujui, maka ditampilkan nama sopir yang mengambil alih penugasan. Kode yang digunakan adalah pada gambar 5.80. Tampilan *activity* saat user ingin menyetujui pengalihan adalah seperti gambar 5.83.

```
private void getInboxData(){
    inboxEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            ModelInbox inbox =
dataSnapshot.getValue(ModelInbox.class);
            String type = inbox.getType();
            String response = inbox.getResponse();
            returnOfGood = inbox.getReturnOfGood();

            ModelAddress modelAdress =
dataSnapshot.child("location").getValue(ModelAddress
.class);
            latLngLocation =new
LatLng(modelAdress.getLatitude(),
modelAdress.getLongitude());
            reason = inbox.getReason();

            Geocoder geocoder;
            List<Address> addresses;
            geocoder = new
Geocoder(getApplicationContext(),
Locale.getDefault());
            try {
                addresses =
geocoder.getFromLocation(latLngLocation.latitude,
latLngLocation.longitude,1);
                location =
addresses.get(0).getAddressLine(0);
                tv_location.setText(location);
            } catch (IOException e) {
                e.printStackTrace();
            }

            tv_Reason.setText(reason);

            if(type.equals("task cancellation")){
                taskID = inbox.getTaskID();
                getTaskData();

rl_Departure.setVisibility(View.GONE);
                isTask=true;
            }
            else if(type.equals("departure
cancellation")){
                departureID =
inbox.getDepartureID();
```

```
                getDepartureData();
                rl_Task.setVisibility(View.GONE);
                isTask=false;
            }

            if(response==null){

textView15.setVisibility(View.VISIBLE);

btn_grant.setVisibility(View.VISIBLE);

btn_decline.setVisibility(View.VISIBLE);
                tv_granted.setVisibility(View.GONE);

tv_rejected.setVisibility(View.GONE);

tv_newDriver.setVisibility(View.GONE);
            }
            else{
                textView15.setVisibility(View.GONE);
                btn_grant.setVisibility(View.GONE);

btn_decline.setVisibility(View.GONE);

tv_checkDriver.setVisibility(View.GONE);

spin_driver.setVisibility(View.GONE);

btn_newDriver.setVisibility(View.GONE);
                if(response.equals("accept")){

tv_granted.setVisibility(View.VISIBLE);

tv_rejected.setVisibility(View.GONE);
                    tv_newDriver.setText("delivery
send to " + inbox.getNewDriver());

tv_newDriver.setVisibility(View.VISIBLE);
                }
                if(response.equals("decline")){

tv_granted.setVisibility(View.GONE);

tv_newDriver.setVisibility(View.GONE);

tv_rejected.setVisibility(View.VISIBLE);
                }
            }
```

```
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataMessage().ref.child(HomeActivity.companyUID)
.child(fleetID).child(messageID).addListenerForSingl
eValueEvent(inboxEventListener);
}

private void getTaskData(){
    taskEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            ModelTask task =
dataSnapshot.getValue(ModelTask.class);
            String customerName =
task.getCustomer();
            customer = customerName;

            LayoutInflater _canvasProduct =
LayoutInflater.from(getApplicationContext());
            View canvasProduct =
_canvasProduct.inflate(R.layout.card_inbox_task,
null, false);
            TextView tv_taskCustomer = (TextView)
canvasProduct.findViewById(R.id.tv_taskCustomer);
            tv_taskCustomer.setText(customerName);

            tv_taskCustomer.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent = new
Intent(getApplicationContext(),
OnGoingDetail.class);
                    intent.putExtra("taskID",
taskID);
                    startActivity(intent);
                }
            });
```

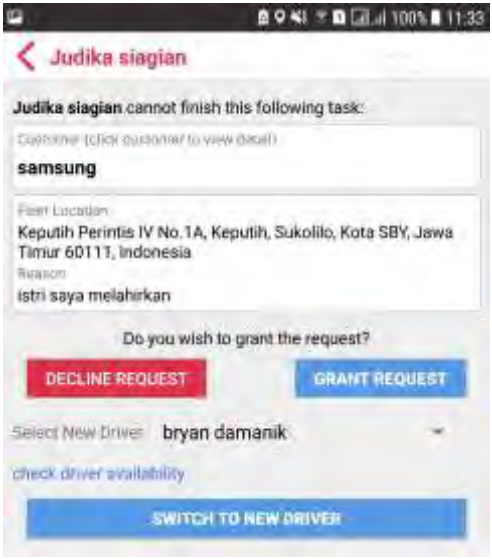
```
            ll_task_taskList.addView(canvasProduct);
            ll_task_taskList.requestLayout();
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSingl
eValueEvent(taskEventListener);
}

private void getDepartureData(){
    departureEventListener = new
ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                String tempTaskID =
postSnapshot.getKey();
                String isComplete =
postSnapshot.child("isTaskCompleted").getValue().toS
tring();
                if(isComplete.equals("false")){
                    taskList.add(tempTaskID);
                    ModelDeparture modelDeparture =
postSnapshot.getValue(ModelDeparture.class);

tv_departureID.setText(modelDeparture.getuID());

tv_returnOfGood.setText(returnOfGood);

                    taskEventListener = new
ValueEventListener() {
                        @Override
                        public void
onDataChange(DataSnapshot dataSnapshot) {
                            ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                            final String _taskID =
dataSnapshot.getKey();
                            String customerName =
task.getCustomer();
```

```
                                if(customer==null){
                                    customer =
customerName;
                                }
                                else{
                                    customer = customer
+ ", " + customerName;
                                }

                                LayoutInflater
_canvasProduct =
LayoutInflater.from(getApplicationContext());
                                View canvaseProduct =
_canvasProduct.inflate(R.layout.card_inbox_task,
null, false);
                                TextView tv_taskCustomer
= (TextView)
canvaseProduct.findViewById(R.id.tv_taskCustomer);

tv_taskCustomer.setText(customerName);


ll_departure_taskList.addView(canvaseProduct);

ll_departure_taskList.requestLayout();


tv_taskCustomer.setOnClickListener(new
View.OnClickListener() {
                                    @Override
                                    public void
onClick(View v) {
                                        Intent intent =
new Intent(getApplicationContext(),
OnGoingDetail.class);

intent.putExtra("taskID", _taskID);

startActivity(intent);
                                    }
                                });

                            }

                            @Override
                            public void
onCancelled(DatabaseError databaseError) {
```

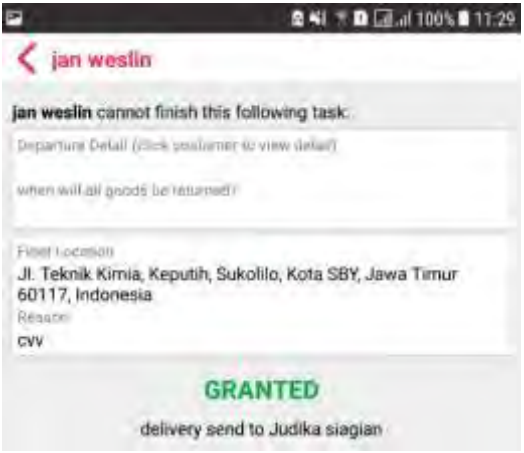
```
                        }
                    };
                    new
FireDataTask().ref.child(tempTaskID).addListenerForS
ingleValueEvent(taskEventListener);
                }
            }

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataDeparture().ref.child(departureID).child("ta
sk").addValueEventListener(departureEventListener);
}
```

Gambar 5.80 US-AM4-3: Mengambil detail data pesan

Gambar 5.81 US-AM4-3: Tampilan detail pesan baru



Gambar 5.82 US-AM4-3: Tampilan pengajuan disetujui



Gambar 5.83 US-AM4-3: Tampilan armada baru telah ditunjuk

Pada gambar 5.81, Saat *user* menekan tombol ***decline request***, *user* hanya memberikan respon menolak lalu menyimpannya dalam *Firebase* untuk dibaca oleh sopir armada. Kode yang digunakan adalah pada gambar 5.84.

```
@OnClick(R.id.btn_decline) public void
declineRequest(){
    new
FireDataMessage().responseDecline(HomeActivity.co
mpanyUID, fleetID, messageID, customer, new
DatabaseReference.CompletionListener() {
        @Override
        public void onComplete(DatabaseError
databaseError, DatabaseReference
databaseReference) {
            ll_task_taskList.removeAllViews();
            getInboxData();
        }
    });
}
```

Gambar 5.84 US-AM4-3: Menolak pengajuan

Saat *user* menekan tombol ***grant request***, aplikasi menampilkan *spinner* untuk memilih sopir baru untuk melanjutkan penugasan.. Setelah *user* memilih, jika yang diajukan adalah 1 *task*, maka ranting *message*→***FleetID*** pada *task* akan diganti pada sopir yang baru, lalu menyimpan respon dalam *Firebase* untuk dibaca oleh sopir armada. Jika yang diajukan adalah 1 *departure*, maka semua *task* yang terdaftar dalam 1 *departure* tersebut akan diganti menjadi sopir yang baru, mengubah status bekerja sopir pada ranting *Fleet*→*FleetID*→***working*** bernilai *false*, menghapus nilai pada ranting *Fleet*→*FleetID*→*departureID*, dan menyimpan respon dalam *Firebase* untuk dibaca oleh sopir armada. Kode yang diguanakan adalah pada Kode 5-85.

```
@OnClick(R.id.btn_newDriver) public void
newDriver(){
    if(isTask==true){
        new FireDataTask().changeDriver(taskID,
newDriverID, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference
databaseReference) {

                new
FireDataMessage().responseAccept(HomeActivity.com
panyUID, fleetID, newDriverName, messageID,
customer, new
DatabaseReference.CompletionListener() {
                    @Override
                    public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {

Toast.makeText(getApplicationContext(),
newDriverName + " selected", Toast.LENGTH_SHORT);

ll_task_taskList.removeAllViews();
                        getInboxData();
                    }
                });
            }
        });

    }
    else if(isTask==false){
        for(int i=0; i<taskList.size();i++){
            final String tempTask =
taskList.get(i);

            new
FireDataTask().changeDriver(tempTask,
newDriverID, new
DatabaseReference.CompletionListener() {
                @Override
                public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                    getInboxData();
```

```
                }
            });

            if(i==taskList.size()-1){
                new
FireDataUserFleet().writeUserStopWorking(fleetID,
new DatabaseReference.CompletionListener() {
                    @Override
                    public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                        new
FireDataUserFleet().ref.child(fleetID).child("dep
artureID").removeValue();
                        new
FireDataDeparture().ref.child(departureID).remove
Value();

                        new
FireDataMessage().responseAccept(HomeActivity.com
panyUID, fleetID, newDriverName, messageID,
customer, new
DatabaseReference.CompletionListener() {
                            @Override
                            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                                new
FireDataTask().ref.child(tempTask).child("departu
reID").removeValue();
                                getInboxData();
                            }
                        });
                    }
                });
            }
        }
    }
}
```

Gambar 5.85 US-AM4-3: Menyetujui pengajuan

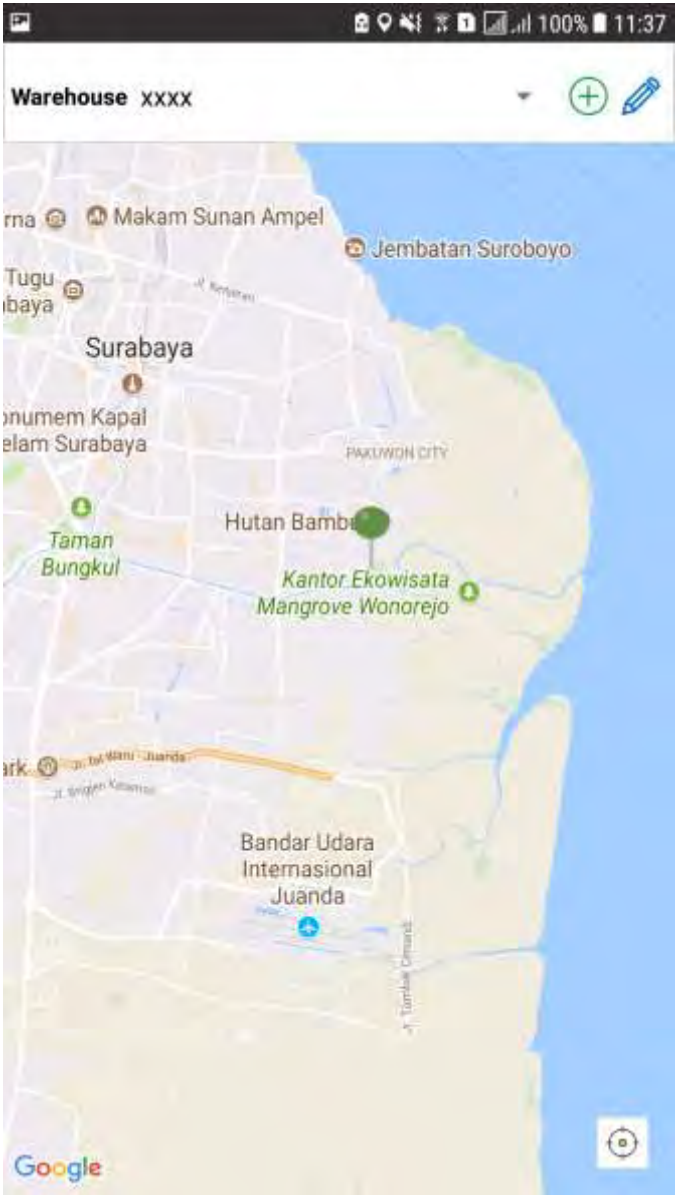
5.2.2.15 US-AM5-1: Membuat, memodifikasi, dan menghapus gudang milik perusahaan

Tampilan untuk mengelola gudang adalah pada gambar 5.87. Pada awal *activity* ini dijalankan, tampilan *default*-nya menggunakan kode pada gambar 5.86.

```
private void setDefaultView(){
    btn_add.setVisibility(View.VISIBLE);
    btn_cancel.setVisibility(View.GONE);
    btn_save.setVisibility(View.GONE);
    rl_editValue.setVisibility(View.GONE);
    btn_addWareHouse.setVisibility(View.GONE);
    btn_delete.setVisibility(View.INVISIBLE);
    ivCenterMap.setVisibility(View.GONE);
    ivSetLocation.setVisibility(View.GONE);
    ivRecenter.setVisibility(View.VISIBLE);
    ivRecenter2.setVisibility(View.GONE);

    if(allDataList.size()==0){
        btn_edit.setVisibility(View.GONE);
    }
    else{
        btn_edit.setVisibility(View.VISIBLE);
    }
}
```

Gambar 5.86 US-AM5-1L: Tampilan default manajemen gudang

Data gudang yang akan ditampilkan diambil dari *Firebase* pada ranting ***warehouse***. Pengambilan data menggunakan *ChildEventListener* karena data gudang akan selalu berubah-ubah pada *activity* ini. Data diambil dan dimasukkan ke *spinner* menggunakan kode pada gambar 5.88.



Gambar 5.87 Tampilan daftar gudang

```
private void getWareHouseData(){
    if(nameList.size()>0) nameList.clear();
    if(allDataList.size()>0)allDataList.clear();
    wareHouseEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {
            ModelWarehouse _modelWarehouse =
dataSnapshot.getValue(ModelWarehouse.class);
            if(_modelWarehouse != null){

if(_modelWarehouse.getCompanyID().equals(user.getUid())
){
                    ModelAddress modelAddress =
dataSnapshot.child(FireDataWareHouse.ADDRESS).getValue(
ModelAddress.class);
                    LatLng latLng = new
LatLng(modelAddress.getLatitude(),
modelAddress.getLongitude());
                    ModelWarehouse modelWarehouse = new
ModelWarehouse(dataSnapshot.getKey(),
_modelWarehouse.getName(), latLng);

                    allDataList.add(modelWarehouse);

nameList.add(modelWarehouse.getName());

                    ArrayAdapter<String> dataAdapter =
new ArrayAdapter<>(getApplicationContext(),

android.R.layout.simple_spinner_dropdown_item,
nameList);

dataAdapter.setDropDownViewResource(android.R.layout.si
mple_spinner_dropdown_item);

spin_wareHouse.setAdapter(dataAdapter);

                    if(allDataList.size()==0){

btn_edit.setVisibility(View.GONE);
                    }
                    else{

btn_edit.setVisibility(View.VISIBLE);
                    }
                }
            }
        }

        @Override
        public void onChildChanged(DataSnapshot
```

```
dataSnapshot, String s) {
            ModelWarehouse _modelWarehouse =
dataSnapshot.getValue(ModelWarehouse.class);
            if(_modelWarehouse != null){

if(_modelWarehouse.getCompanyID().equals(user.getUid())
){
                    ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.cla
ss);
                    LatLng latLng = new
LatLng(modelAddress.getLatitude(),
modelAddress.getLongitude());
                    ModelWarehouse modelWarehouse = new
ModelWarehouse(dataSnapshot.getKey(),
_modelWarehouse.getName(), latLng);

                    allDataList.remove(wareHouseData);
                    nameList.remove(wareHouseName);

                    allDataList.add(modelWarehouse);

nameList.add(modelWarehouse.getName());

                    ArrayAdapter<String> dataAdapter =
new ArrayAdapter<>(getApplicationContext(),

android.R.layout.simple_spinner_dropdown_item,
nameList);

dataAdapter.setDropDownViewResource(android.R.layout.si
mple_spinner_dropdown_item);

spin_wareHouse.setAdapter(dataAdapter);

                    if(allDataList.size()==0){

btn_edit.setVisibility(View.GONE);
                    }
                    else{

btn_edit.setVisibility(View.VISIBLE);
                    }
                }
            }
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            ModelWarehouse modelWarehouse =
dataSnapshot.getValue(ModelWarehouse.class);
```

```
            if(modelWarehouse != null){

if(modelWarehouse.getCompanyID().equals(user.getUid()))
{
                    allDataList.remove(modelWarehouse);

nameList.remove(modelWarehouse.getName());

                    ArrayAdapter<String> dataAdapter =
new ArrayAdapter<>(getApplicationContext(),

android.R.layout.simple_spinner_dropdown_item,
nameList);

dataAdapter.setDropDownViewResource(android.R.layout.si
mple_spinner_dropdown_item);

spin_wareHouse.setAdapter(dataAdapter);

                    if(allDataList.size()==0){

btn_edit.setVisibility(View.GONE);
                    }
                    else{

btn_edit.setVisibility(View.VISIBLE);
                    }
                }
            }
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataWareHouse().ref.addChildEventListener(wareHouse
EventListener);
    spin_wareHouse.setOnItemSelectedListener(this);
}
```

Gambar 5.88 US-AM5-1: Mengambil data gudang

Aktivitas *Company_Warehouse* memungkinkan *user* untuk:

1. Menambah gudang baru
   Untuk menambah gudang baru, tombol yang ditekan adalah *btn_addNewWarehouse*. *User* harus mengisi nama gudang baru dan alamatnya. Ujung pin alamat berada di tengah layar, jadi untuk mengambil kordinat lokasi tengah layar digunakan kode pada gambar 5.89. Tampilan untuk menambah gudang adalah pada Gambar 8-93.

```
@OnClick(R.id.ivSetLocation) public void
setLocation(){
    VisibleRegion visibleRegion =
mMap.getProjection().getVisibleRegion();
    Point x =
mMap.getProjection().toScreenLocation(visibleReg
ion.farRight);
    Point y =
mMap.getProjection().toScreenLocation(visibleReg
ion.nearLeft);
    Point centerPoint = new Point(x.x / 2, y.y /
2);
    centerFromPoint =
mMap.getProjection().fromScreenLocation(centerPo
int);

    if(wareHouseMarker != null){
        wareHouseMarker.remove();
    }
    wareHouseMarker = mMap.addMarker(new
MarkerOptions().position(centerFromPoint).title(
wareHouseName).

icon(BitmapDescriptorFactory.fromBitmap(resizeBi
tmap("map_warehouse", 59, 100))));
    Log.d("tsss","center from point: " +
centerFromPoint.toString());

    newAddress = new
LatLng(centerFromPoint.latitude,
centerFromPoint.longitude);
}
```

Gambar 5.89 US-AM5-1: Mengambil kordinat peta

Setelah mengisi nama dan menentukan lokasi, aplikasi akan mengisikan data gudang baru di *Firebase* pada ranting ***warehouse***. Kode yang digunakan pada gambar 5.90.

```
@OnClick(R.id.btn_addWareHouse) public void
addNewWareHouse(){
    final String newName =
et_name.getText().toString();
    if(TextUtils.isEmpty(newName)){
        Toast.makeText(this,"please enter your
warehouse name", Toast.LENGTH_SHORT).show();
        return;
    }
    if(newAddress == null){
        Toast.makeText(this,"please select the
location", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataWareHouse().writeWareHouse(user.getUid()
, newName, newAddress, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference
databaseReference) {
                if(databaseError == null){
                    setDefaultView();
                    newAddress=null;

Toast.makeText(getApplicationContext(),"new
warehouse successfully added",
Toast.LENGTH_SHORT).show();
                }
                else{
                    Log.d("Tsss", "error
creating new warehouse: " +
databaseError.getDetails());
                }
            }
        });
    }
}
```

Gambar 5.90 US-AM5-1: Menambah gudang baru

2. Memperbaharui data gudang
   Untuk memperbaharui gudang, *user* pertama sekali menekan tombol *btn_edit* untuk memunculkan *layout edit*. Setelah itu nama yang tersimpan akan tertulis di *EditText* nama gudang. Jika *user* ingin mengganti nama gudang, *user* tinggal menghapus dan mengganti nama baru pada *EditText* tersebut, jika ingin mengganti kordinat alamat, *user* tinggal memilih alamat seperti membuat gudang baru. Setelah diganti, *user* menekan tombol *btn_save* untuk menyimpan data baru. Kode yang digunakan adalah pada gambar 5.92. Tampilan untuk memperbaharui nama gudang adalah pada gambar 5.94 dan untuk memperbaharui lokasi gudang adalah pada gambar 5.95.

3. Menghapus gudang
   Jika ingin menghapus gudang, *user* tinggal menekan tombol *btn_delete* dan aplikasi akan menghapus gudang sesuai dengan posisi yang terpilih di *spinner* kemudian menampilkan Actvitas ini pada tampilan default seperti gambar 5.87.. Kode yang digunakan adalah pada gambar 5.91.

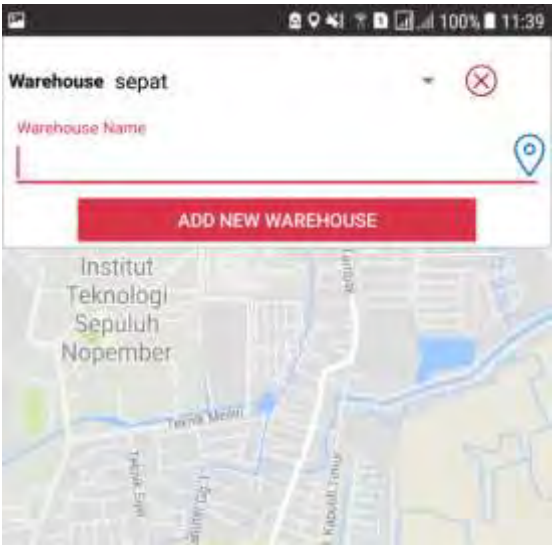
```
@OnClick(R.id.btn_delete) public void
deleteWareHouse(){
    new
FireDataWareHouse().ref.child(wareHouseID).removeVa
lue();
    setDefaultView();
}
```

Gambar 5.91 US-AM5-1: Menghapus gudang

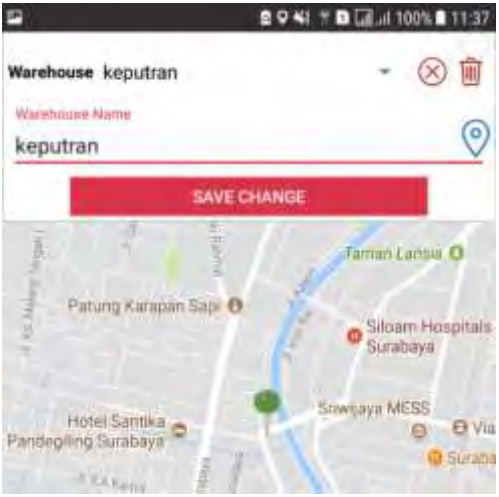
```
@OnClick(R.id.btn_save) public void
saveWareHouse(){
    final String newName =
et_name.getText().toString();
    if(TextUtils.isEmpty(newName) && newAddress ==
null){
        setDefaultView();
    }
    if(!TextUtils.isEmpty(newName) && newAddress ==
null){
        new
FireDataWareHouse().writeName(wareHouseID,
user.getUid(), newName , new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError==null){
                    setDefaultView();
                    newAddress=null;
                }
                else{
                    Log.d("Tsss", "error saving
edit name: " + databaseError.getDetails());
                }
            }
        });
    }
    if(TextUtils.isEmpty(newName) && newAddress !=
null){
        new
FireDataWareHouse().writeAddress(wareHouseID,
user.getUid(), newAddress, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError==null){
                    setDefaultView();
                    newAddress=null;
                }
                else{
                    Log.d("Tsss", "error saving
edit address: " + databaseError.getDetails());
                }
            }
        });
    }
    if(!TextUtils.isEmpty(newName) && newAddress!=
null){
```

```
        new
FireDataWareHouse().writeAll(wareHouseID,
user.getUid(), newName, newAddress, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError==null){
                    setDefaultView();
                    newAddress=null;
                }
                else{
                    Log.d("Tsss", "error saving
edit name and address: " +
databaseError.getDetails());
                }
            }
        });
    }
}
```

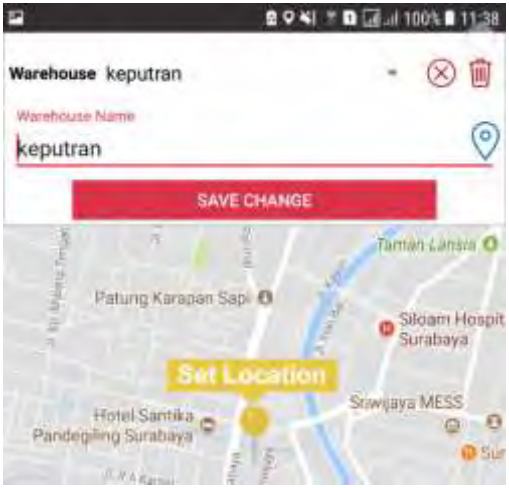
Gambar 5.92 US-AM5-1: Memperbaharui gudang



Gambar 5.93 Tampilan menambah gudang baru



Gambar 5.94 Tampilan mengubah nama gudang



Gambar 5.95 Tampilan mengubah lokasi gudang

5.2.2.16 US-AM6-1: *Reset Password*

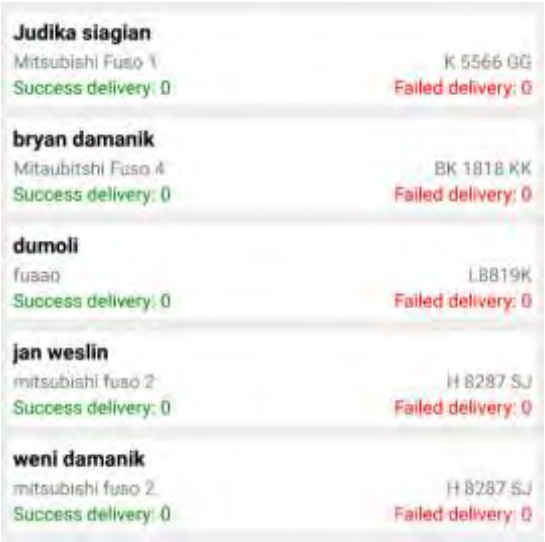
Fitur reset password ada terdapat dalam Aktivitas *SignInActivity*. Saat user lupa atas Password miliknya, user bisa mengajukan reset password yang nantinya akan dikelola langsung dari Firebase. Untuk bisa melakukannya, user harus tahu email yang ia gunakan, kemudian Firebase akan mengirimkan email untuk mengubah password. Kode yang digunakan adalah pada gambar 5.96.

```
@OnClick(R.id.reset_hint) public void reset(){
    email = et_id.getText().toString();
    if(TextUtils.isEmpty(email)){
        Toast.makeText(this, "Please enter your email
without password", Toast.LENGTH_SHORT).show();
        return;
    }
    else{

mAuth.sendPasswordResetEmail(email).addOnCompleteListen
er(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void>
task) {
                Toast.makeText(getApplicationContext(),
"Check your email", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Gambar 5.96 US-AM6-1: Reset password

5.2.2.17 US-AM7-1: Melihat kinerja tiap armada

User dapat melihat kinerja tiap armada yang dimilikinya berdasarkan penyelesaian tugas masing-masing armada. User dapat melihat jumlah tugas yang berhasil diselesaikan armada dan yang belum berhasil diselesaikan pada daftar armada yang dimilikinya pada fragment FleetAccountList. Bentuknya adalah seperti gambar 5.97.



Gambar 5.97 US-AM7-1: Daftar armada perusahaan

Data sukses dan tidak sukses tersebut diambil dari ranting Firebase yakni: Fleet→FleetID→countSuccess untuk jumlah yang sukses dan Fleet→FleetID→countFailed untuk jumlah yang gagal. Nilai pada countSuccess dan countFailed tersebut bertambah saat armada mengirimkan laporan penyelesaian tugasnya. Apabila berhasil maka nilia countSuccess bertambah, apabila gagal, maka nilai countFailed yang bertambah.

5.3 *Smartphone* Sopir Armada Perusahaan

5.3.1 Lingkungan Aplikasi

Tabel 5.2 Lingkungan aplikasi fleemo driver

Virtual Machine	
Nama perangkat	Nexus 5
Processor	Multi-Core 4 CPUs
Memory	1536 MB
Sistem Operasi	Android API 24 x86 Nougat

Resolusi	5.2” 1080x1920 xxhdpi
Perangkat Nyata	
Nama perangkat	Samsung A5 2016
Processor	Samsung Exynos Octa 7580 1.60GHz
Memory	1827MB
Sistem Operasi	Android API 24 x86 Nougat
Resolusi	5.2” 1080x1920 xxhdpi

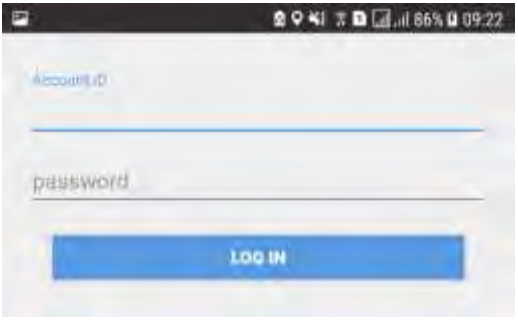
5.3.2 Fungsi-Fungsi Aplikasi

Pada bagian ini dijelaskan proses implementasi setiap fungsi aplikasi dengan potongan kode yang digunakan. Ide yang dipakai adalah **Android Studio 3.0.1.**

Berikut adalah fungsi-fungsi yang dimiliki oleh *Fleemo Driver*, sesuai dengan User Story pada Tabel 4-4.

5.3.2.1 US-AS1-1: *Sign in*

SignInActivity merupakan *activity* yang memungkinkan aplikasi untuk *sign in* dan mengakses seluruh data yang diizinkan pada *firebase*. gambar 5.98 berikut adalah tampilan *SignInActivity*:



Gambar 5.98 US-AS1-1: Tampilan SignInActivity

Pada aplikasi *Fleemo Driver*, hanya *email* yang terdaftar sebagai armada atau pada *database* berada pada ranting ***Fleet* saja yang boleh masuk**, sedangkan *email* yang berada pada ranting ***user* tidak boleh**. Akun armada juga ada 2 jenis, yakni yang belum dan sudah pernah *login*. Keduanya sudah terdaftar di *Firebase database*, tapi yang belum pernah *login* masih belum terdaftar di *Firebase* Authentication, sehingga yang belum pernah *login* akan didaftarkan terlebih dahulu di Authentication kemudian masuk ke *activity* utama. Pada awal *activity* ini, aplikasi mengakses data *firebase* untuk mengambil *email* apa saja yang terdaftar pada ranting *Fleet* dan juga keterangan *sign up*-nya lalu memasukkannya ke dalam *list* menggunakan kode pada gambar 5.99.

Setelah *user* memasukkan *email* dan *password*, aplikasi akan mengecek apakah *email* dan *password* sudah dimasukkan. Apabila sudah masuk, kemudian dicek apakah *email* cocok dengan yang ada di dalam *list*. Apabila cocok, kemudian dicek apakah sudah pernah *sign up* atau belum. Jika sudah pernah *sign up*, nilai booleannya adalah *true*. Kode yang digunakan adalah kode 5-100. Apabila nilai *signUp* adalah *true*, maka kode yang dijalankan untuk *sign in* adalah Kode 5-101. Dan Apabila nilai *signUp* adalah *false*, maka urutan perintah dimulai dari mengambil seluruh data yang ada dalam *Firebase* dengan ranting *Fleet*→*FleetID* sesuai dengan *email* yang diketik. Lalu melakukan perintah membuat *user* baru di *Firebase* Authentication. Jika berhasil, data armada yang tersimpan tadi dihapus lalu dibuat yang baru dengan *FleetID* yang sama dengan uID authentication. Setelah berhasil barulah masuk ke *activity* utama. Kode yang dijalankan untuk *sign up* adalah pada gambar 5.102.

```
private void getEmail(){
    emailEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot)
{
            for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                String key = postSnapshot.getKey();
                String tempEmail =
postSnapshot.child(FireDataUser.DRIVER_EMAIL).getValue().toS
tring();
                boolean tempisSignUp =
postSnapshot.child(FireDataUser.IS_SIGNUP).getValue(Boolean.
class);
                Log.d("tsss", "tmpEmail: " + tempEmail + " :
tempisSignUp: " + tempisSignUp);
                ModelFleet fleet;
                if(tempisSignUp==true) {
                    fleet = new ModelFleet(key, tempEmail,
tempisSignUp);
                }
                else{
                    ModelFleet _fleet =
postSnapshot.getValue(ModelFleet.class);
                    ModelAddress modelAddress =
postSnapshot.child(FireDataUser.DRIVER_ADDRESS).getValue(Mod
elAddress.class);
                    String companyID =
_fleet.getCompanyID();
                    String fullname = _fleet.getFullname();
                    String email = _fleet.getEmail();
                    String password = _fleet.getPassword();
                    String license = _fleet.getLicense();
                    String phone = _fleet.getPhone();
                    double lat = modelAddress.getLatitude();
                    double lng =
modelAddress.getLongitude();
                    String gender = _fleet.getGender();
                    String birthdate =
_fleet.getBirthdate();
                    String truckName  =
_fleet.getTruckName();
                    String truckPlate  =
_fleet.getTruckPlate();
                    fleet = new ModelFleet(key, companyID,
fullname, email, password, license, phone, lat, lng, gender,
birthdate, truckName, truckPlate);
                }
                fleetList.add(fleet);
            }
        }
```

```
        @Override
        public void onCancelled(DatabaseError databaseError)
{

        }
    };
    new
FireDataUser().ref.addValueEventListener(emailEventListener)
;
}
```

Gambar 5.99 US-AS-1: Mengambil email dan keterangan sign-up user

```
@OnClick(R.id.btn_login) public void btn_login(){
    email = et_id.getText().toString();
    password = et_password.getText().toString();
    if(TextUtils.isEmpty(email)){
        Toast.makeText(this, "Please enter your ID",
Toast.LENGTH_SHORT).show();
        return;
    }
    if(TextUtils.isEmpty(password)){
        Toast.makeText(this, "Please enter your Password",
Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        for(int i=0; i<fleetList.size(); i++){
            if(email.equals(fleetList.get(i).getEmail())){
                if(fleetList.get(i).getSignUp()==false){
                    //Sign Up
                }
                else if(fleetList.get(i).getSignUp()==true){
                    //Sign In
                }
            }
            else{
                Toast.makeText(this, "Account is not
registered", Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Gambar 5.100 US-AS1-1: cek email dan sign up

```
try{
    mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                token =
FirebaseInstanceId.getInstance().getToken();
                new FireDataUser().writeUserPassword(uID,
password, new DatabaseReference.CompletionListener() {
                    @Override
                    public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                        startActivity(new
Intent(SignInActivity.this, HomeActivity.class));
                        finish();
                    }
                });
            }
            else{
                Toast.makeText(getApplicationContext(),
"Password incorrect", Toast.LENGTH_SHORT).show();
            }
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(getApplicationContext(),
e.getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}
catch(Throwable t) {
    t.printStackTrace();
    AlertDialog alertDialog = new
AlertDialog.Builder(this).create();
    alertDialog.setTitle("Error");
    alertDialog.setMessage("Application running on Android 4.4
and above");
    alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int
which) {
                    dialog.dismiss();
                }
            });
    alertDialog.show();
    return;
}
```

Gambar 5.101 US-AS1-1: Sign in

```
tmpPassword = fleetList.get(i).getPassword();

if(tmpPassword.equals(password)){
    companyID = fleetList.get(i).getCompanyID();
    fullname =fleetList.get(i).getFullname();
    license = fleetList.get(i).getLicense();
    phoneNumber = fleetList.get(i).getPhone();
    address = new LatLng(fleetList.get(i).getLat(),
fleetList.get(i).getLang());
    gender = fleetList.get(i).getGender();
    birthDate = fleetList.get(i).getBirthdate();
    truckName = fleetList.get(i).getTruckName();
    truckPlate = fleetList.get(i).getTruckPlate();

    try{

mAuth.createUserWithEmailAndPassword(email,password).addOn
CompleteListener(this, new
OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull
Task<AuthResult> task) {
                if(task.isSuccessful()){
                    Toast.makeText(SignInActivity.this,
"Account Registered successfully",
Toast.LENGTH_SHORT).show();

                    new
FireDataUser().writeUser(mAuth.getCurrentUser().getUid(),
companyID, fullname, email, password, license,
phoneNumber,
                            address, gender, birthDate,
truckName, truckPlate, new
DatabaseReference.CompletionListener() {
                                @Override
                                public void
onComplete(DatabaseError databaseError, DatabaseReference
databaseReference) {
                                    if(databaseError ==
null){
                                        //delete old file
user
                                        new
FireDataUser().writeUserToken(FirebaseAuth.getInstance().g
etUid(), token, new DatabaseReference.CompletionListener()
{
                                            @Override
                                            public void
onComplete(DatabaseError databaseError, DatabaseReference
databaseReference) {

startActivity(new Intent(SignInActivity.this,
```

```
HomeActivity.class));
                                                    finish();
                                                    new
FireDataUser().ref.child(uID).removeValue();
                                                }
                                            });
                                        }
                                        else{

Toast.makeText(SignInActivity.this, "Recreating user
error", Toast.LENGTH_SHORT).show();
                                        }
                                    }
                                });
                        }
                        else {
                            Toast.makeText(SignInActivity.this,
"Authentication error. Contact administrator",
Toast.LENGTH_SHORT).show();
                            Log.d("tsss", "Registration error: " +
task.getException());
                        }
                    }
                });
        }
        catch(Throwable t){
            t.printStackTrace();
            AlertDialog alertDialog = new
AlertDialog.Builder(this).create();
            alertDialog.setTitle("Error");
            alertDialog.setMessage("Application running on
Android 4.4 and above");
            alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL,
"OK",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int which) {
                            dialog.dismiss();
                        }
                    });
            alertDialog.show();
            Toast.makeText(this, "Application running on
Android 4.4 and above", Toast.LENGTH_LONG).show();
            return;
        }
    }
    else{
        Toast.makeText(getApplicationContext(), "Password
incorrect", Toast.LENGTH_SHORT).show();
    }
```

Gambar 5.102 US-AS1-1: Sign up

5.3.2.2 US-AS1-2: Menerima dan melihat tugas yang diberikan perusahaan

Tugas-tugas yang diberikan perusahaan untuk dikerjakan *user* dapat dilihat pada *fragment NotFinished_Task_List* yang disajikan dalam barisan *card*. Tiap *card* memiliki konten pelanggan, alamat, tipe pekerjaan (*regular task* dan *rework task*), dan *CheckBox* untuk memilih tugas yang akan diantarkan. Semua data tersedia dalam 1 ranting *Task* di *Firebase*, jadi hanya perlu mengambil 1 data saja. Data yang diambil hanya yang bernilai ***false*** di ranting *Task→TaskID→isCompleted*. Pengambilan data *task* menggunakan kode pada gambar 5.103. Pengambilan data menggunakan ChildEventListener karena data penugasan dapat selalu berubah-ubah (tugas bertambah, tugas selesai, tugas dihapus). Dengan ChildEventListener, perubahan data dicek secara terus-menerus.

Data tiap tugas dimasukkan ke dalam *HashMap taskMap*, alamat penugasan ke dalam *HashMap* addressMap. *HashMap* ini nantinya dimasukkan ke dalam *adapater* yang memungkinkan 1 key pada *HashMap* ditampilkan dalam 1 *card*. *Adapter* yang akan ditampilkan dalam RecyclerView recyclerview_notFinished. Apabila tidak ada tugas yang tersimpan, RecyclerView tidak muncul dan sebagai gantinya memunculkan *TextView* yang menjelaskan tidak ada penugasan yang terekam. Kode yang digunakan adalah pada gambar 5.104. Kemudian untuk konfigurasi pada *AdapterView* digunakan kode pada gambar 5.105.

```
private void initTaskList() {
    taskEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {

if(fleetID.equals(dataSnapshot.child("fleetID").getValu
e().toString())){

if(dataSnapshot.child("isCompleted").getValue().toStrin
g().equals("false")){
                    final String key =
dataSnapshot.getKey();
                    final ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    final ModelAddress address =
dataSnapshot.child("address").getValue(ModelAddress.cla
ss);
                    taskMap.put(key, task);
                    addressMap.put(key, address);
                    mAdapter.notifyDataSetChanged();
                    checkEmpty();
                }
            }
        }

        @Override
        public void onChildChanged(DataSnapshot
dataSnapshot, String s) {

if(fleetID.equals(dataSnapshot.child("fleetID").getValu
e().toString())){

if(dataSnapshot.child("isCompleted").getValue().toStrin
g().equals("false")){
                    final String key =
dataSnapshot.getKey();
                    final ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    final ModelAddress address =
dataSnapshot.child("address").getValue(ModelAddress.cla
ss);
                    taskMap.put(key, task);
                    addressMap.put(key, address);
                    mAdapter.notifyDataSetChanged();
                    checkEmpty();
                }
            }
            else
if(!fleetID.equals(dataSnapshot.child("fleetID").getVal
ue().toString())){
```

```
if(dataSnapshot.child("isCompleted").getValue().toStrin
g().equals("false")){
                    final String key =
dataSnapshot.getKey();
                    taskMap.remove(key);
                    addressMap.remove(key);
                    mAdapter.notifyDataSetChanged();
                    checkEmpty();
                }
            }
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            String key = dataSnapshot.getKey();
            taskMap.remove(key);
            addressMap.remove(key);
            mAdapter.notifyDataSetChanged();
            checkEmpty();
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.addChildEventListener(taskEventListe
ner);
}
```

Gambar 5.103 US-AS1-2: Mengambil data task

Adapter tersebut menggunakan *card_not_finish_task* dengan tampilan pada gambar 5.106. *Card* tersebut memiliki CheckBox agar *user* dapat memilih tugas-tugas mana saja yang ia hendak kerjakan dalam sekali pengantaran.

```
private void initRecycler(){
    mAdapter = new NotFinishedTaskAdapter(taskMap,
addressMap, getActivity());
    RecyclerView.LayoutManager mLayoutManager = new
LinearLayoutManager(getActivity());

recyclerview_notfinished.setLayoutManager(mLayoutManage
r);
    recyclerview_notfinished.setItemAnimator(new
DefaultItemAnimator());
    recyclerview_notfinished.setAdapter(mAdapter);

    checkEmpty();
}

private void checkEmpty(){
    if(taskMap.size()>0){

recyclerview_notfinished.setVisibility(View.VISIBLE);

tv_emptyNotFinishedTask.setVisibility(View.GONE);
    }
    else {

recyclerview_notfinished.setVisibility(View.GONE);

tv_emptyNotFinishedTask.setVisibility(View.VISIBLE);
    }
}
```

Gambar 5.104 US-AS1-2: Set adapter pada recycler view

Jika salah satu *card* di klik, maka mausuk ke *activity NotFinished_Task_Detail*. Untuk memuat detail penugasan, digunakan kode pada gambar 5.107. Tampilan detail penugasan dapat dilihat pada gambar 5.108. Pada Aktivitas tersebut ditampilkan nama pelanggan, alamat pelanggan, jenis penugasan, lokasi gudang untuk mengambil barang yang harus diantarkan, detail barang apa saja yang harus diantarkan, dan tombol untuk mengajukan pengalihan tugas.

```
private Map<String, ModelTask> taskMap = new
HashMap<>();
private Map<String, ModelAddress> addressMap = new
LinkedHashMap<>();
private Map<String, ModelTask> departureMap = new
HashMap<>();
private OnItemClickListener mOnItemClickListener;
private Activity activity;

public interface OnItemClickListener {
    void onItemClick(View view, ModelTask obj, int
position, String key);
}

public void setOnItemClickListener(final
OnItemClickListener mItemClickListener) {
    this.mOnItemClickListener = mItemClickListener;
}

public class MyViewHolder extends
RecyclerView.ViewHolder{

    @BindView(R.id.tv_notfinished_customer) TextView
notfinished_customer;
    @BindView(R.id.tv_notfinished_address) TextView
notfinished_address;
    @BindView(R.id.tv_notfinished_type) TextView
tv_notfinished_type;
    @BindView(R.id.cb_SelectTask) CheckBox
cb_SelectTask;
    @BindView(R.id.rlCard) RelativeLayout rlCard;

    public MyViewHolder(View view) {
        super(view);
        ButterKnife.bind(this, view);
    }
}

public NotFinishedTaskAdapter(Map<String, ModelTask>
productMap, Map<String, ModelAddress> addressMap,
Activity activity ){
    this.taskMap = productMap;
    this.addressMap = addressMap;
    this.activity = activity;
}

@Override
public MyViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
    View itemView=
LayoutInflater.from(parent.getContext())
```

```
.inflate(R.layout.card_not_finish_task,parent,false);

    return new MyViewHolder(itemView);
}

@Override
public void onBindViewHolder(MyViewHolder holder, int
position) {
    final List<String> keyList = new
ArrayList<>(taskMap.keySet());
    final List<ModelTask> taskList = new
ArrayList<>(taskMap.values());
    final List<ModelAddress> addressList = new
ArrayList<>(addressMap.values());
    final String key = keyList.get(position);
    final ModelTask task = taskList.get(position);
    final ModelAddress modelAddress =
addressList.get(position);
    String _address=null;

    Geocoder geocoder;
    List<Address> addresses;
    geocoder = new
Geocoder(activity.getApplicationContext(),
Locale.getDefault());
    try {
        addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
        _address = addresses.get(0).getAddressLine(0);
    } catch (IOException e) {
        e.printStackTrace();
    }



holder.notfinished_customer.setText(task.getCustomer())
;
    holder.notfinished_address.setText(_address);

holder.tv_notfinished_type.setText(task.getWorkType());

    holder.cb_SelectTask.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {
            if(isChecked){
                departureMap.put(key, task);
                Log.d("tsss", "selected :"+ key);
            }
            else{
```

```
                departureMap.remove(key);
                Log.d("tsss", "removed :"+key);
            }
        }
    });

    holder.rlCard.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new
Intent(v.getContext().getApplicationContext(),
NotFinished_Task_Detail.class);
            intent.putExtra("taskID", key);
            v.getContext().startActivity(intent);
        }
    });
}

@Override
public int getItemCount() {
    return taskMap.size();
}

public Map<String, ModelTask> getProductMap(){
    return this.departureMap;
}
```

Gambar 5.105 US-AS1-2: NotFinishedTaskAdapter

Jika ingin melakukan pengantaran, pada *activity NotFinished_Task_List*, tandai tugas-tugas yang akan diantar pada *checkbox* yang tersedia, lalu menekan tombol *btn_check*. Yang berwarna biru berbentuk segitiga pada gambar 5.106. Kode yang digunakan adalah pada gambar 5.109.

Gambar 5.106 US-AS1-2: card_not_finish_task

```
private void loadProduct(){
    singleEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            final ModelTask modelTask=
dataSnapshot.getValue(ModelTask.class);
            final ModelAddress modelAddress =
dataSnapshot.child(FireDataTask.ADDRESS).getValue(Model
Address.class);
            final ModelAddress modelWareHouse =
dataSnapshot.child(FireDataTask.WAREHOUSELOC).getValue(
ModelAddress.class);

            Geocoder geocoder;
            List<Address> addresses1;
            List<Address> addresses2;
            geocoder = new
Geocoder(getApplicationContext(), Locale.getDefault());
            try {
                addresses1 =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                addresses2 =
geocoder.getFromLocation(modelWareHouse.getLatitude(),
modelWareHouse.getLongitude(),1);
                addressCustomer =
addresses1.get(0).getAddressLine(0);
                addressWareHouse =
addresses2.get(0).getAddressLine(0);
            } catch (IOException e) {
                e.printStackTrace();
            }

            valueEventListener = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot) {

tv_notfinished_address.setText(addressCustomer);

tv_notfinished_customer.setText(modelTask.getCustomer()
);

tv_taskType.setText(modelTask.getWorkType());

tv_wareHouseName.setText((modelTask.getWareHouseName())
);

tv_wareHouseAddress.setText(addressWareHouse);
```

```
                    for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                        final ModelMessage modelMessage
= postSnapshot.getValue(ModelMessage.class);
                        Log.d("tsss","message: "+
modelMessage.getItem());
                        LayoutInflater _canvasProduct =
LayoutInflater.from(getApplicationContext());
                        View canvasProduct =
_canvasProduct.inflate(R.layout.card_item_not_finished,
null, false);
                        TextView
tv_notfinished_productname = (TextView)
canvasProduct.findViewById(R.id.tv_notfinished_productn
ame);
                        TextView
tv_notfinished_productamount = (TextView)
canvasProduct.findViewById(R.id.tv_notfinished_producta
mount);


tv_notfinished_productname.setText(modelMessage.getItem
());

tv_notfinished_productamount.setText(modelMessage.getTa
rget());


rl_product.addView(canvasProduct);
                        rl_product.requestLayout();
                    }
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataTask().ref.child(taskID).child("message").addVa
lueEventListener(valueEventListener);
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSingleVa
```

```
lueEvent(singleEventListener);
}
```

Gambar 5.107 US-AS1-2: Memuat detail penugasan

.

Gambar 5.108 US-AS1-2: Tampilan detail not_finish_task

```
@OnClick(R.id.btn_check) public void startDelivery(){
    final List<String> key = new
ArrayList<>(mAdapter.getProductMap().keySet());
    if(key.size()>0){
        for(int i=0; i<key.size(); i++) {
            String task = key.get(i);

            Map<String, Object> objectMap = new
HashMap<>();
            objectMap.put(FireDataDeparture.TASK_ID,
task);

objectMap.put(FireDataDeparture.TASKCOMPLETED, false);

            departureMap.put(task, objectMap);
            if(i == key.size()-1){
                new
FireDataDeparture().writeDeparture(HomeActivity.company
ID, fleetID, departureMap, new
DatabaseReference.CompletionListener() {
                    @Override
                    public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                        final String departureID =
databaseReference.getKey();

                        new
FireDataUser().writeUserStartWorking(fleetID,
departureID,new DatabaseReference.CompletionListener()
{
                            @Override
                            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {

if(databaseError==null){

Toast.makeText(getActivity(), "Starting delivery
product", Toast.LENGTH_SHORT).show();

                                    for(int j=0;
j<key.size(); j++){
                                        String _task =
key.get(j);
                                        new
FireDataTask().writeTaskDeparture(_task, departureID,
new DatabaseReference.CompletionListener() {
                                            @Override
                                            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
```

```
                                                    // Go
to Job Intent, that could not interrupt
                                                    Intent
intent = new Intent(getContext(), TaskOngoing.class);

intent.putExtra("departureID", departureID);

startActivity(intent);
                                                }
                                            });
                                        }
                                    }
                                }
                            });
                        }
                    });

                }
            }
        }
        else{
                Toast.makeText(getContext(), "Select at
least 1 task to deliver", Toast.LENGTH_SHORT).show();
            }
}
```

Gambar 5.109 US-AS1-2: Mulai melakukan pengantaran

5.3.2.3 US-AS1-3: Mengajukan pengalihan tugas ke sopir lain

Saat sopir memiliki halangan untuk mengerjakan tugas dengan alasan yang logis, sopir dapat mengajukan pengalihan tugas ke sopir lain pada perusahaan. Pada halaman detail tugas yang belum selesai, terdapat tombol untuk membatalkan penugasan pada posisi paling bawah dengan keterangan "*Request Task Cancellation*". Saat mengklik tombol tersebut, masuk ke *activity CancelTask* dengan extra nilai *taskID*. Kode yang digunakan adalah pada gambar 5.110.

```
@OnClick(R.id.btn_cancel) public void btn_cancel(){
    Intent intent = new
Intent(NotFinished_Task_Detail.this,CancelTask.class
);
    intent.putExtra("taskID", taskID);
    startActivity(intent);
}
```

Gambar 5.110 US-AS1--3: Memulai *activity* CancelTask

CancelTask hanya berisi 1 *EditText* untuk tempat *user* mengisikan alasannya. gambar 5.111 berikut adalah tampilan *CancelTask*:

Gambar 5.111 US-AM1-3: Tampilan CancelTask

Pada *activity CancelTask*, *user* perlu mengaktifkan GPS agar aplikasi bisa mendapatkan lokasi terkini sopir saat mengirimkan penajuan. Setelah *user* mengisikan alasan pengalihan tugas, *user* mengklik Submit. Aplikasi akan mengambil lokasi terkini, lalu menyimpan pesan tersebut dalam *Firebase* dengan ranting ***message***. Kode yang digunakan adalah pada gambar 5.112.

```
private void getCurrentLocation(){
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }
    Location location =
LocationServices.FusedLocationApi.getLastLocation(googl
eApiClient);
    if (location != null) {
        fleetLoc = new LatLng(location.getLatitude(),
location.getLongitude());
    }
}

@OnClick(R.id.btn_cancelsubmit) public void
btn_cancelsubmit(){
    getCurrentLocation();
    reason = et_reason.getText().toString();
    if(fleetLoc==null){
        Toast.makeText(this, "Cannot Retrieve current
location. Try again", Toast.LENGTH_LONG).show();
    }
    else if(TextUtils.isEmpty(reason)){
        Toast.makeText(this, "Fill your reason",
Toast.LENGTH_SHORT).show();
    }
    else{
        new
FireDataMessage().writeCancelTask(HomeActivity.companyI
D, HomeActivity.fleetUID, taskID, reason, fleetLoc, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                Toast.makeText(getApplicationContext(),
"Your request will be considered",
Toast.LENGTH_LONG).show();
                startActivity(new
Intent(getApplicationContext(), HomeActivity.class));
                finish();
            }
        });
    }

}
```

Gambar 5.112 US-AM1-3: Mengirimkan pesan ajuan pengalihan tugas

5.3.2.4 US-AS2-1: Melihat tugas yang sudah diselesaikan

Tugas-tugas yang sudah diselesaikan oleh *user* dapat dilihat pada *fragment* Finished_*Task_List* yang disajikan dalam barisan *card*. Tiap *card* memiliki konten pelanggan, *departureID*, alamat, kategori keberhasilan (*successfully delivered*, *unsuccessfully delivered*, *rework with same Fleet*, dan *rework with other Fleet*), dan tipe pekerjaan (*regular task* dan *rework task*). Semua data tersedia dalam 1 ranting ***Task*** di *Firebase*. Data yang diambil adalah semua yang bernilai **selain *false*** di ranting *Task→TaskID→isCompleted* karena nilai false diletakkan pada *fragment NotFinished_Task_List*. Pengambilan data *task* menggunakan kode pada gambar 5.113.

Data tiap tugas dimasukkan ke dalam *HashMap taskMap*, alamat penugasan ke dalam *HashMap* addressMap. Apabila tidak ada tugas yang tersimpan pada *taskMap*, *RecyclerView* tidak muncul dan sebagai gantinya memunculkan *TextView* yang menjelaskan tidak ada penugasan yang terekam. Kode yang digunakan untuk memasukkan *HashMap* ke *adapter* adalah pada gambar 5.114. Kemudian Konfugurasi pada *AdapterView* adalah pada gambar 5.115. *Adapter* tersebut menggunakan *card_finish_task* dengan tampilan pada gambar 5.116.

Jika salah satu *card* diklik, maka masuk ke *activity Finished_Task_Detail* yang memiliki 2 *fragment*, yakni deskripsi penugasan bernama *Finished_Description* dan *Finished_Route*. Untuk memuat detail penugasan pada *Finished_Description* digunakan kode pada gambar 5.117, dan ditampilkan seperti gambar 5.118.

```
private void initTaskList(){
    childEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {

if(fleetID.equals(dataSnapshot.child("fleetID").getV
alue().toString())){

if(!dataSnapshot.child("isCompleted").getValue().toS
tring().equals("false")){
                    final String key =
dataSnapshot.getKey();
                    final ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    final ModelAddress address =
dataSnapshot.child("address").getValue(ModelAddress.
class);
                    taskMap.put(key, task);
                    addressMap.put(key, address);
                    mAdapter.notifyDataSetChanged();
                    checkEmpty();
                }
            }
        }

        @Override
        public void onChildChanged(DataSnapshot
dataSnapshot, String s) {

if(fleetID.equals(dataSnapshot.child("fleetID").getV
alue().toString())){

if(dataSnapshot.child("isCompleted").getValue().toSt
ring().equals("true")){
                    final String key =
dataSnapshot.getKey();
                    final ModelTask task =
dataSnapshot.getValue(ModelTask.class);
                    final ModelAddress address =
dataSnapshot.child("address").getValue(ModelAddress.
class);
                    taskMap.put(key, task);
                    addressMap.put(key, address);
                    mAdapter.notifyDataSetChanged();
                    checkEmpty();
                }
            }
```

```
            else
if(!fleetID.equals(dataSnapshot.child("fleetID").get
Value().toString())){

if(dataSnapshot.child("isCompleted").getValue().toSt
ring().equals("true")){
                    final String key =
dataSnapshot.getKey();
                    taskMap.remove(key);
                    addressMap.remove(key);
                    mAdapter.notifyDataSetChanged();
                    checkEmpty();
                }
            }
        }
        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            String key = dataSnapshot.getKey();
            taskMap.remove(key);
            addressMap.remove(key);
            mAdapter.notifyDataSetChanged();
            checkEmpty();
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.addChildEventListener(childEventL
istener);
}
```

Gambar 5.113 US-AS1-3: Mengambil data task

```
private void initRecycler(){
    mAdapter = new FinishedTaskAdapter(taskMap,
addressMap, getActivity());
    RecyclerView.LayoutManager mLayoutManager =
new LinearLayoutManager(getActivity());

recyclerview_finished.setLayoutManager(mLayoutMan
ager);
    recyclerview_finished.setItemAnimator(new
DefaultItemAnimator());
    recyclerview_finished.setAdapter(mAdapter);

    checkEmpty();
}

private void checkEmpty(){
    if(taskMap.size()>0){

recyclerview_finished.setVisibility(View.VISIBLE)
;

tv_emptyFinishedTask.setVisibility(View.GONE);
    }
    else {

recyclerview_finished.setVisibility(View.GONE);

tv_emptyFinishedTask.setVisibility(View.VISIBLE);
    }
}
```

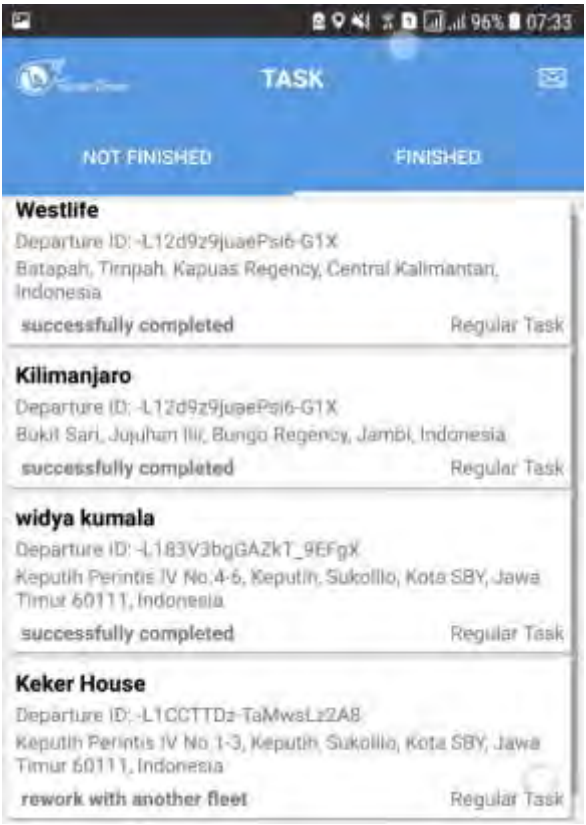
Gambar 5.114 US-AS1-3: Set adapter recycler view

```
private Map<String, ModelTask> productMap = new
HashMap<>();
private Map<String, ModelAddress> addressMap = new
LinkedHashMap<>();
private Activity activity;
private NotFinishedTaskAdapter.OnItemClickListener
mOnItemClickListener;

public interface OnItemClickListener {
    void onItemClick(View view, ModelTask obj, int
position, String key);
}

public void setOnItemClickListener(final
NotFinishedTaskAdapter.OnItemClickListener
mItemClickListener) {
    this.mOnItemClickListener = mItemClickListener;
}

public class MyViewHolder extends
RecyclerView.ViewHolder{

    @BindView(R.id.tv_finished_customer) TextView
tv_finished_customer;
    @BindView(R.id.tv_finished_departureID) TextView
tv_finished_departureID;
    @BindView(R.id.tv_finished_address) TextView
tv_finished_address;
    @BindView(R.id.tv_notfinished_type) TextView
tv_notfinished_type;
    @BindView(R.id.tv_notfinished_isCompleted) TextView
tv_notfinished_isCompleted;
    @BindView(R.id.rlCard) RelativeLayout rlCard;

    public MyViewHolder(View view) {
        super(view);
        ButterKnife.bind(this, view);
    }
}
public FinishedTaskAdapter(Map<String, ModelTask>
productMap, Map<String, ModelAddress> addressMap,
Activity activity){
    this.productMap = productMap;
    this.addressMap = addressMap;
    this.activity = activity;
}

@Override
public MyViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
    View itemview =
LayoutInflater.from(parent.getContext()).
```

```
inflate(R.layout.card_finish_task,parent,false);

    return new MyViewHolder(itemview);
}

@Override
public void onBindViewHolder(MyViewHolder holder, int
position) {
    final List<String> keyList = new
ArrayList<>(productMap.keySet());
    final List<ModelTask> taskList = new
ArrayList<>(productMap.values());
    final List<ModelAddress> addressList = new
ArrayList<>(addressMap.values());
    final String key = keyList.get(position);
    final ModelTask task = taskList.get(position);
    final ModelAddress modelAddress =
addressList.get(position);
    String _address=null;

    Geocoder geocoder;
    List<Address> addresses;
    geocoder = new
Geocoder(activity.getApplicationContext(),
Locale.getDefault());
    try {
        addresses =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
        _address = addresses.get(0).getAddressLine(0);
    } catch (IOException e) {
        e.printStackTrace();
    }


holder.tv_finished_customer.setText(task.getCustomer())
;
    holder.tv_finished_departureID.setText("Departure
ID: " + task.getDepartureID());
    holder.tv_finished_address.setText(_address);

holder.tv_notfinished_type.setText(task.getWorkType());

holder.tv_notfinished_isCompleted.setText(task.isComple
ted());

    holder.rlCard.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new
```

```
Intent(v.getContext().getApplicationContext(),
Finished_Task_Detail.class);
            intent.putExtra("taskID", key);
            v.getContext().startActivity(intent);
        }
    });
}

@Override
public int getItemCount() {
    return productMap.size();
}
```

Gambar 5.115 US-ASI-3: FinishedTaskAdapter



Gambar 5.116 US-AS1-3: card_finish_task

```
private void loadproduct(){
    singleEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            final ModelTask task =
dataSnapshot.getValue(ModelTask.class);
            final ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.cla
ss);
            final ModelAddress modelWareHouse =
dataSnapshot.child("warehouseLoc").getValue(ModelAddres
s.class);
            valueEventListener = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot) {
                    Geocoder geocoder;
                    List<Address> addresses1;
                    List<Address> addresses2;
                    geocoder = new
Geocoder(getContext(), Locale.getDefault());
                    try {
                        addresses1 =
geocoder.getFromLocation(modelAddress.getLatitude(),
modelAddress.getLongitude(),1);
                        addresses2 =
geocoder.getFromLocation(modelWareHouse.getLatitude(),
modelWareHouse.getLongitude(),1);
                        addressCustomer =
addresses1.get(0).getAddressLine(0);
                        addressWareHouse =
addresses2.get(0).getAddressLine(0);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }



tv_finished_customer.setText(task.getCustomer());

tv_finished_address.setText(addressCustomer);

tv_ongoing_departureID.setText(task.getDepartureID());

tv_taskType.setText(task.getWorkType());

tv_completionStatus.setText(task.isCompleted());

tv_wareHouseName.setText(task.getWareHouseName());
```

```
tv_wareHouseAddress.setText(addressWareHouse);

                    for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                        ModelMessage message =
postSnapshot.getValue(ModelMessage.class);
                        LayoutInflater _canvasProduct =
LayoutInflater.from(getContext());
                        View canvasProduct =
_canvasProduct.inflate(R.layout.card_item_finished,
null, false);
                        TextView
tv_finished_productname = (TextView)
canvasProduct.findViewById(R.id.tv_finished_productname
);
                        TextView
tv_finished_producttarget = (TextView)
canvasProduct.findViewById(R.id.tv_finished_producttarg
et);
                        TextView
tv_finished_productactual = (TextView)
canvasProduct.findViewById(R.id.tv_finished_productactu
al);


tv_finished_productname.setText(message.getItem());

tv_finished_producttarget.setText(message.getTarget());

tv_finished_productactual.setText(message.getActual());


rl_finished_product.addView(canvasProduct);

rl_finished_product.requestLayout();
                    }
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataTask().ref.child(taskID).child("message").addVa
lueEventListener(valueEventListener);
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {
```

```
        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSingleVa
lueEvent(singleEventListener);
}
```

Gambar 5.117 US-AS1-3: Memuat detail penugasan

Gambar 5.118 US-AS1-3: Tampilan detail deskripsi penugasan

Kemudian untuk memuat kordinat-kordinat pada *Route_Detail*, digunakan kode pada gambar 5.119. Tampilan detail rute penugasan adalah pada gambar 5.120.

```
private void getDatabase(){
    taskEventListener = new ValueEventListener()
{
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            ModelTask task =
dataSnapshot.getValue(ModelTask.class);
            ModelAddress modelAddress =
dataSnapshot.child(FireDataTask.ADDRESS).getValue
(ModelAddress.class);
            ModelAddress modelWareHouse =
dataSnapshot.child(FireDataTask.WAREHOUSELOC).get
Value(ModelAddress.class);
            latLngCustomer = new
LatLng(modelAddress.getLatitude(),
modelAddress.getLongitude());
            latLngWareHouse = new
LatLng(modelWareHouse.getLatitude(),
modelWareHouse.getLongitude());

            mMap.clear();
            customerMarker = mMap.addMarker(new
MarkerOptions().position(latLngCustomer).title(ta
sk.getCustomer()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBit
map("map_customer", 52, 100))));
            wareHouseMarker = mMap.addMarker(new
MarkerOptions().position(latLngWareHouse).title(t
ask.getWareHouseName()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBit
map("map_warehouse", 52, 100))));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom
(latLngCustomer, 12));


if(dataSnapshot.child(FireDataTask.LOCATION).getV
alue() != null){
```

```
                trackEventListener = new
ValueEventListener() {
                    @Override
                    public void
onDataChange(DataSnapshot dataSnapshot) {
                        for(DataSnapshot
postSnapshot : dataSnapshot.getChildren()){
                            ModelAddress _track =
postSnapshot.getValue(ModelAddress.class);
                            LatLng track = new
LatLng(_track.getLatitude(),
_track.getLongitude());
                            mMap.addMarker(new
MarkerOptions().position(track).

icon(BitmapDescriptorFactory.fromBitmap(resizeBit
map("map_fleet_old_location", 30, 30))));
                        }
                    }

                    @Override
                    public void
onCancelled(DatabaseError databaseError) {

                    }
                };
                new
FireDataTask().ref.child(taskID).child(FireDataTa
sk.LOCATION).addValueEventListener(trackEventList
ener);
            }
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSi
ngleValueEvent(taskEventListener);
}
```

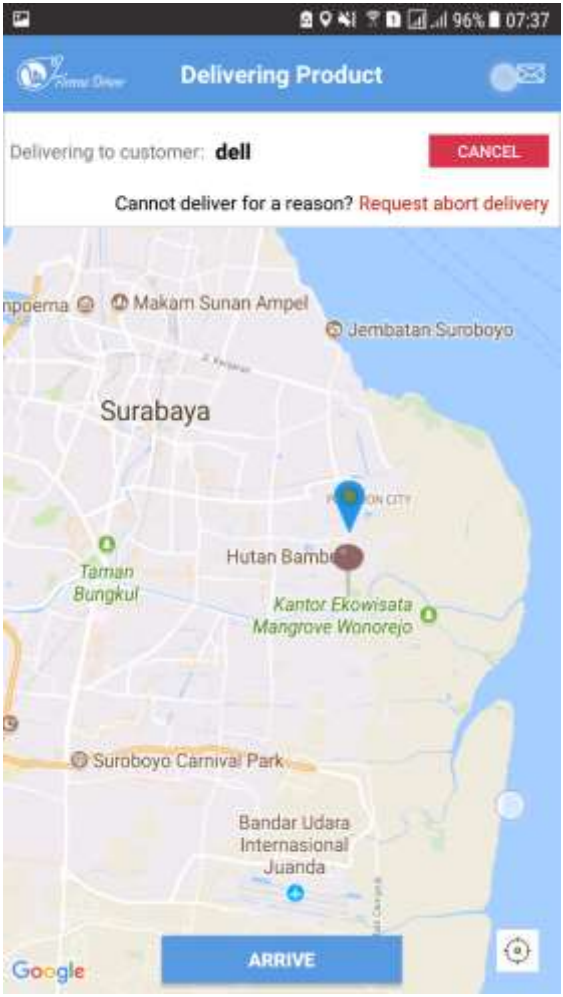
Gambar 5.119 US-AS1-3: Memuat detail kordinat lokasi

Gambar 5.120 US-AS1-3: Tampilan detail rute penugasan

5.3.2.5 US-AS3-1: Menyimpan riwayat lokasi saat melaksanakan pengiriman barang

Riwayat lokasi disimpan saat melakukan pengantaran atau dengan kata lain pada ranting *Fleet→FleetID→departureID* memiliki nilai dari tugasnya sekarang. gambar 5.121 dan gambar 5.122 adalah tampilan *activity* pengantaran barang bernama ***TaskOngoing***.

Gambar 5.121 US-AS3-1: Tampilan TaskOngoing – memilih tugas



Gambar 5.122 US-AS3-1: Tampilan TaskOngoing – mengantarkan tugas

Dari beberapa tugas yang terdaftar dalam 1 *departureID*, pengantaran hanya bisa dilakukan satu per satu. Saat 1 tugas sudah dipilih dan pengantaran dimulai barulah aplikasi

menyimpan riwayat lokasi pengiriman barang. Kode yang digunakan adalah pada gambar 5.123.

```
private void getCurrentLocation() {
    LocationListener mLocationListener = new
LocationListener() {
        @Override
        public void onLocationChanged(Location
location) {
            fleetLatLng = new
LatLng(location.getLatitude(),
location.getLongitude());
            new
FireDataUser().writeUserLocation(HomeActivity.fle
etUID, fleetLatLng, new
DatabaseReference.CompletionListener() {
                @Override
                public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                    Log.d("tsss", "Location
Update: "+fleetLatLng);
                }
            });

            if(fleetMarker != null)
fleetMarker.remove();
            fleetMarker = mMap.addMarker(new
MarkerOptions().position(fleetLatLng).title("Flee
t")

.icon(BitmapDescriptorFactory.fromBitmap(resizeBi
tmap("map_fleet_current_location", 42, 80))));

mMap.moveCamera(CameraUpdateFactory.newLatLng(fle
etLatLng));

            if(taskID != null){
                if(fleetLatLng!=null){
                    new
FireDataTask().writeTaskLocation(taskID,
fleetLatLng, new
DatabaseReference.CompletionListener() {
                        @Override
                        public void
```

```
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {

                        }
                    });
                }
            }
        }

        @Override
        public void onStatusChanged(String
provider, int status, Bundle extras) {

        }

        @Override
        public void onProviderEnabled(String
provider) {

        }

        @Override
        public void onProviderDisabled(String
provider) {

        }
    };

    LocationManager mLocationManager =
(LocationManager)
getSystemService(LOCATION_SERVICE);
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED
            &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {

mLocationManager.requestLocationUpdates(LocationM
anager.GPS_PROVIDER, 2000,20, mLocationListener);
    }
}
```

Gambar 5.123 US-AS3-1: Mengambil lokasi terkini

Perubahan lokasi akan dicatat setiap perpindahan minimal 20m atau minimal 2 detik. Setiap perpndahan mencapai batal minimal, dibuat *marker* di peta, lalu kordinatnya disimpan di *firebase* pada ranting *task→taskID→location→cordinateID*. Penyimpanan ini dilakukan *realtime* untuk mengantisipasi aplikasi yang tertutup dan agar lokasi terkini langsung bisa diakses oleh perusahaan. Apabila aplikasi tiba-tiba tertutup, maka untuk mengembalikan *marker* dari lokasi yang sudah dilalui, data diambil dari *Firebase* dengan kode pada gambar 5.124.

```
private void setfleetTrack(){

    trackEventListener = new ValueEventListener()
{
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            for (DataSnapshot postSnaphot :
dataSnapshot.getChildren()){
                ModelAddress _track =
postSnaphot.getValue(ModelAddress.class);
                LatLng track = new
LatLng(_track.getLatitude(),
_track.getLongitude());

                mMap.addMarker(new
MarkerOptions().position(track).

icon(BitmapDescriptorFactory.fromBitmap(resizeBit
map("map_fleet_old_location", 25, 25))));
            }
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataTask().ref.child(taskID).child(FireDataTa
```

```
sk.LOCATION).addValueEventListener(trackEventList
ener);
}
```

Gambar 5.124 US-AS3-1: Mengambil lokasi tersimpan dari Firebase

5.3.2.6 US-AS3-2: Mengisi *form* penyelesaian tugas

Form penyelesaian tugas bisa dijalankan saat pengantaran tugas atau saat membuka *activity* ***TaskOngoing***. Ketika sampai di lokasi, *user* menekan tombol *btn_finish* untuk masuk ke *activity CompletionActivity*. Kode yang digunakan adalah pada gambar 5.125.

```
@OnClick(R.id.btn_finish) public void
finishTask(){
    //When Task Done. Go to Completion Page.
    Intent intent = new
Intent(getApplicationContext(),
CompletionActivity.class);
    intent.putExtra("taskID", taskID);
    intent.putExtra("departureID", departureID);
    startActivity(intent);
}
```

Gambar 5.125 US-AS3-2: Menjalankan *activity* CompletionActivity

CompletionActivity memuat data-data dari *task* terkait, seperti nama pelanggan, alamat, dan semua produk yang harus diantarkan kemudian menampilkannya. Juga memuat data token namun tidak ditampilkan. Nilai ini akan menjadi acuan apakah token saat penyelesaian tugas sudah sesuai dengan yang tersimpan di *database*. Kode yang digunakan adalah pada gambar 5.126.

Aplikasi juga mengambil data kinerja user yang tersimpan karena saat penyelesaian tugas akan ditambahkan. Data diambil dari Ranting fleet→fleetID→countSuccess dan fleet→fleetID →countFailed. Jika tugas berhasil diselesaikan, maka nilai

countSuccess yang akan bertambah. Kode yang digunakan adalah pada gambar 5.127. Data ini diperlukan perusahaan agar tahu berapa banyak tugas yang berhasil dan gagal diselesaikan oleh tiap armada miliknya.

```
public void loadTask(){
    taskEventListener = new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            final ModelTask task=
dataSnapshot.getValue(ModelTask.class);
            final ModelAddress modelAddress =
dataSnapshot.child("address").getValue(ModelAddress.cla
ss);
            latitude = modelAddress.getLatitude();
            longitude = modelAddress.getLongitude();

            Geocoder geocoder;
            List<Address> addresses;
            geocoder = new
Geocoder(getApplicationContext(), Locale.getDefault());
            try {
                addresses =
geocoder.getFromLocation(latitude, longitude,1);
                address =
addresses.get(0).getAddressLine(0);
            } catch (IOException e) {
e.printStackTrace();            }

            messageEventListener = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot) {
                    token = task.getToken();

tv_completion_customer.setText(task.getCustomer());

tv_completion_address.setText(address);

                    for(DataSnapshot postSnapshot :
dataSnapshot.getChildren()){
                        //Load Product From Database
                        final ModelMessage modelMessage
= postSnapshot.getValue(ModelMessage.class);
                        LayoutInflater _canvasProduct =
LayoutInflater.from(getApplicationContext());
                        View canvasProduct =
```

```
_canvasProduct.inflate(R.layout.card_item_arrives,
null, false);
                        TextView
tv_completion_productname = (TextView)
canvasProduct.findViewById(R.id.tv_completion_productna
me);
                        TextView
tv_completion_producttarget = (TextView)
canvasProduct.findViewById(R.id.tv_completion_producttа
rget);;
                        EditText
et_completion_productName = (EditText)
canvasProduct.findViewById(R.id.et_completion_productam
ount);


tv_completion_productname.setText(modelMessage.getItem(
));

tv_completion_producttarget.setText(modelMessage.getTar
get());


rl_completion_product.addView(canvasProduct);

rl_completion_product.requestLayout();

                        //Save Product to Array
                        listSize++;
                        ModelMessage _product = new
ModelMessage(listSize, tv_completion_productname,
tv_completion_producttarget,
et_completion_productName);
                        productList.add(_product);

                    }
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataTask().ref.child(taskID).child("message").addVa
lueEventListener(messageEventListener);
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {
```

```
        }
    };
    new
FireDataTask().ref.child(taskID).addListenerForSingleVa
lueEvent(taskEventListener);
}
```

Gambar 5.126 US-AS3-2: Mengambil detail data task dari Firebase

```
public void fleetPerform(){
    fleetEventListener = new ValueEventListener()
{
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            countSuccess =
dataSnapshot.child(FireDataUser.COUNT_SUCCESS).ge
tValue(Integer.class);
            countFailed =
dataSnapshot.child(FireDataUser.COUNT_FAILED).get
Value(Integer.class);
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataUser().ref.child(fleetID).addListenerForS
ingleValueEvent(fleetEventListener);
}
```

Gambar 5.127 US-AS3-2: Mengambil data kinerja user

Daftar barang yang harus sampai pada pelanggan ditampilkan dalam *card* yang berisi nama produk, *target* (barang yang seharusnya sampai), dan *EditText actual* (barang yang sampai) seperti gambar 5.128.

Bukan *user*, melainkan pelanggan yang harus mengisikan nama produk, token, lalu mengirimkannya pada *Firebase*. Aplikasi akan mengecek apakah ada *actual* yang kurang dari jumlah

target. Apabila ada, maka nilai *completionID* menjadi "*unsuccessfully delivered*" sementara jika semua jumlahnya sama nilainya adalah "*successfully completed*". Dengan kedua ID ini, *task* ini akan masuk ke daftar *task* yang sudah selesai. Selain itu, aplikasi juga mengirimkan jumlah hasil pekerjaan yang berhasil/gagal oleh user dari nilai countSuccess atau countFailed. Kode yang digunakan adalah pada gambar 5.129.

Gambar 5.128 US-AS3-2: Tampilan Completion_Activity

```
private void sendCompletion(){
    new
FireDataTask().writeCompletedDelivery(taskID,
completeMessage, productMap, new
DatabaseReference.CompletionListener() {
        @Override
        public void onComplete(DatabaseError
databaseError, DatabaseReference
databaseReference) {

            new
FireDataDeparture().writeCompletedDelivery(depart
ureID, taskID, completeMessage, new
DatabaseReference.CompletionListener() {
                @Override
                public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                    checkDepartureComplete();
                }
            });
        }
    });
}

@OnClick(R.id.btn_completion_submit) public void
submit(){

if(et_token.getText().toString().equals(token)){
        for(int i = 0; i< productList.size();
i++){
            String id =
productList.get(i).getId() + "";
            String item =
productList.get(i).getTvItem().getText() + "";
            String target =
productList.get(i).getTvTarget().getText() + "";
            String actual =
productList.get(i).getEtActual().getText() + "";


if(Integer.parseInt(actual)<Integer.parseInt(targ
et)){
                completeMessage = "unsuccessfully
delivered";
            }
```

```
            Map<String, Object> objectMap = new
HashMap<>();

objectMap.put(FireDataTask.MESSAGE_VALUE_ITEM,
item);

objectMap.put(FireDataTask.MESSAGE_VALUE_TARGET,
target);

objectMap.put(FireDataTask.MESSAGE_VALUE_ACTUAL,
actual);
            productMap.put(id, objectMap);

            if(i == productList.size()-1){

if(completeMessage.equals("unsuccessfully
delivered")){
                    countFailed++;
                    new
FireDataUser().writeUserCountFailed(fleetID,
countFailed, new
DatabaseReference.CompletionListener() {
                        @Override
                        public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                            sendCompletion();
                        }
                    });
                }
                else
if(completeMessage.equals("successfully
completed")){
                    countSuccess++;
                    new
FireDataUser().writeUserCountSuccess(fleetID,
countSuccess, new
DatabaseReference.CompletionListener() {
                        @Override
                        public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                            sendCompletion();
                        }
                    });
```

```
                }
            }
        }
    }
    else{
        Toast.makeText(CompletionActivity.this,
"token incorrect", Toast.LENGTH_SHORT).show();
        return;
    }
}
```

Gambar 5.129 US-AS3-2: Mengirim data penyelesaian tugas ke Firebase

5.3.2.7 US-AS3-3: Mengajukan pembatalan pengantaran tugas yang sedang dikerjakan

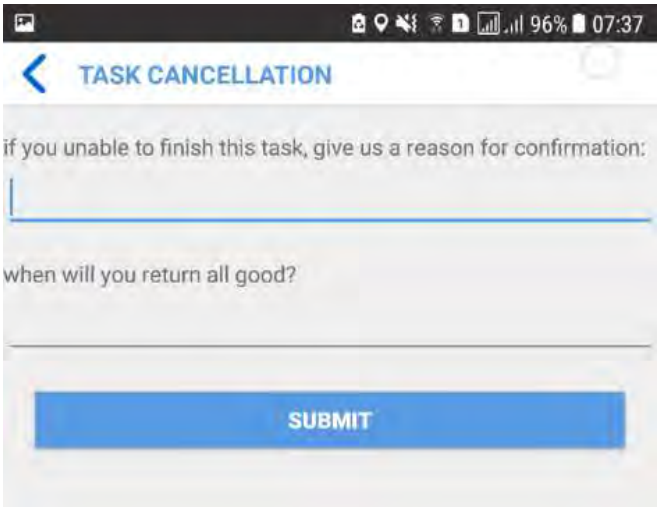
Saat dalam perjalanan mengantar tugas, *user* bisa saja mengalami kendala yang mengharuskan ia mengajukan pengalihan penugasan ke sopir lain. Untuk mengajukannya, *user* mengklik *TextView* "***Request abort delivery***" dalam *activity* TaskOngoing, kemudian masuk ke *activity* ***CancelDeparture***. Jika mengajukan pembatan pengantaran tugas, maka semua *task* yang ada dalam 1 *departureID* akan dibatalkan. Kode yang digunakan adalah pada gambar 5.130.

```
@OnClick(R.id.tv_abort) public void abort(){
    Intent intent = new Intent(this,
CancelDeparture.class);
    intent.putExtra("departureID", departureID);
    startActivity(intent);
}
```

Gambar 5.130 US-AS3-3: Memulai *activity* CancelDeparture

CancelDeparture berisi 2 *EditText* untuk tempat *user* mengisikan alasannya dan penjelelasan kapan barang yang sudah dibawanya akan dikembalikan. Tampilan CancelDeparture ada pada gambar 5.131.

Pada *activity* Cancel*Departure*, *user* perlu mengaktifkan GPS agar aplikasi bisa mendapatkan lokasi terkini sopir saat mengirimkan pengajuan. Setelah *user* mengisikan alasan pengalihan tugas dan kapan barang dikembalikan, *user* mengklik *Submit*. Aplikasi akan mengambil lokasi terkini, lalu menyimpan pesan tersebut dalam *Firebase* dengan ranting *message*. Kode yang digunakan adalah pada gambar 5.132.



Gambar 5.131 US_AS3-3: Tampilan CancelDeparture

5.3.2.8 US-AS3-4: Menerima konfirmasi tugas yang belum selesai untuk dilanjutkan atau dialihkan

Tidak ada pemberitahuan khusus berupa notifikasi untuk memberitahu apakah *user* tersebut yang melanjutkan tugas yang belum diselesaikannya atau *user* lain. Hal itu bisa diketahui saat *user* melihat daftar tugas yang belum dikerjakan. Ada tugas baru dengan tipe *Rework Task*.

```
@OnClick(R.id.btn_cancelsubmit) public void
btn_cancelsubmit(){
    getCurrentLocation();
    reason = et_reason.getText().toString();
    returnOfGood =
et_returnOfGood.getText().toString();
    if(fleetLoc==null){
        Toast.makeText(this, "Cannot Retrieve
current location. Try again",
Toast.LENGTH_LONG).show();
    }
    else if(TextUtils.isEmpty(reason) ||
TextUtils.isEmpty(returnOfGood)){
        Toast.makeText(this, "Fill the form",
Toast.LENGTH_SHORT).show();
    }
    else{
        new
FireDataMessage().writeCancelDeparture(HomeActivity.
companyID, HomeActivity.fleetUID, departureID,
reason, fleetLoc,
                returnOfGood,new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{

Toast.makeText(getApplicationContext(), "Your
request will be considered",
Toast.LENGTH_LONG).show();
                finish();
            }
        });
    }

}
```

Gambar 5.132 US-AS3-3: Mengirimkan pesan ajuan pengalihan

5.3.2.9 US-AS3-5: Menerima pemberitahuan pembatalan penugasan

Saat *user* mengajukan pengalihan tugas, perusahaan akan merespon baik dengan penolakan atau diizinkan. Semua respon tersebut akan ditampilkan pada *activity* ***MessageActivity***. Pesan disajikan dalam barisan *card*. Tiap *card* memiliki konten yakni Nama perusahaan, jenis pengalihan (task cancellation, *departure cancellation*), dan respon dari perusahaan. Apabila disetujui, pesan akan berwarna hijau, sedangkan bila ditolak akan berwarna merah. Tidak ada detail dari tiap pesan, sehingga *card* tidak bisa diklik. Jadi untuk memberitahu sistem apakah pesan sudah dibaca atau tidak, ada tombol "*I understand*". Data diambil dari ranting *Message→companyID→FleetID→ messageID*. Apabila ranting response dalam *messageID* memiliki nilai, maka *message* ditampilkan. Kode yang digunakan adalah pada gambar 5.133. Dalam MessageActivity, selain respon atas pengalihan penugasan, ada juga pesan pembatalan transaksi agar tidak diantarkan oleh user.

Data tiap pesan dimasukan ke dalam *HashMap message*Map. *HashMap* ini nantinya akan dimasukkan ke dalam *adapter* yang memungkinkan 1 key pada *HashMap* ditampilkan dalam 1 *card*. *Adapter* akan ditampilkan dalam *RecyclerView recyclerview_message*. Kode yang digunakan adalah pada gambar 5.134. Kemudian untuk konfigurasi pada *AdapterView* digunakan kode pada gambar 5.135.

Adapter tersebut memungkinkan agar konten yang disampaikan ke *user* terlihat berbeda. Respon penolakan dibuat berwarna merah, respon menyetujui dibuat berwarna hijau, dan pesan bahwa penugasan telah dihapus karena transaksi dibatalkan dibuat berwarna biru. *Adapter* tersebut menggunakan *card_message* dengan tampilan pada gambar 5.136.

```
private void initMessageList(){
    inboxEventListener = new ChildEventListener() {
        @Override
        public void onChildAdded(final DataSnapshot
dataSnapshot1, String s) {
            final ModelInbox _message =
dataSnapshot1.getValue(ModelInbox.class);
            final String key = dataSnapshot1.getKey();
            companyEventListener = new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot) {
                    String companyName =
dataSnapshot.child("fullname").getValue().toString();
                    String type = _message.getType();
                    final String response =
_message.getResponse();
                    final String customer =
_message.getCustomer();
                    if(response != null){
                        boolean isRead =
dataSnapshot1.child(FireDataMessage.IS_READ).getValue(B
oolean.class);
                        Bitmap profil = null;
                        messageMap.put(key, new
ModelInbox(HomeActivity.companyID, companyName, profil,
type, customer, response, isRead));

messageAdapter.notifyDataSetChanged();
                        checkEmpty();
                    }
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataCompany().ref.child(HomeActivity.companyID).add
ListenerForSingleValueEvent(companyEventListener);
        }

        @Override
        public void onChildChanged(final DataSnapshot
dataSnapshot1, String s) {
            final ModelInbox _message =
dataSnapshot1.getValue(ModelInbox.class);
            final String key = dataSnapshot1.getKey();
            companyEventListener = new
```

```
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot
dataSnapshot) {
                    String companyName =
dataSnapshot.child("fullname").getValue().toString();
                    String type = _message.getType();
                    final String response =
_message.getResponse();
                    final String customer =
_message.getCustomer();
                    if(response != null){
                        boolean isRead =
dataSnapshot1.child(FireDataMessage.IS_READ).getValue(B
oolean.class);
                        Bitmap profil = null;
                        messageMap.put(key, new
ModelInbox(HomeActivity.companyID, companyName, profil,
type, customer, response, isRead));

messageAdapter.notifyDataSetChanged();
                        checkEmpty();
                    }
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            };
            new
FireDataCompany().ref.child(HomeActivity.companyID).add
ListenerForSingleValueEvent(companyEventListener);
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            String key = dataSnapshot.getKey();
            messageMap.remove(key);
            messageAdapter.notifyDataSetChanged();
            checkEmpty();
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

        }

        @Override
```

```
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataMessage().ref.child(HomeActivity.companyID).chi
ld(HomeActivity.fleetUID).addChildEventListener(inboxEv
entListener);
}
```

Gambar 5.133 US-AS3-4: Mengambil data message dari Firebase

```
private void initRecycler(){
    messageAdapter = new MessageAdapter(messageMap);
    RecyclerView.LayoutManager layoutManager = new
LinearLayoutManager(this);

recyclerview_message.setLayoutManager(layoutManager)
;
    recyclerview_message.setItemAnimator(new
DefaultItemAnimator());
    recyclerview_message.setAdapter(messageAdapter);

    checkEmpty();
}

private void checkEmpty(){
    if(messageMap.size()>0){

recyclerview_message.setVisibility(View.VISIBLE);
        tv_emptyMessage.setVisibility(View.GONE);
    }
    else {

recyclerview_message.setVisibility(View.GONE);
        tv_emptyMessage.setVisibility(View.VISIBLE);
    }
}
```

Gambar 5.134 US-AS3-3: set Adapter pada recycler view

```
private Map<String, ModelInbox> messageMap = new
LinkedHashMap<>();

public class MyViewHolder extends
RecyclerView.ViewHolder{

    @BindView(R.id.iv_profil) ImageView iv_profil;
    @BindView(R.id.tv_sender) TextView tv_sender;
    @BindView(R.id.tv_type) TextView tv_type;
    @BindView(R.id.tv_content) TextView tv_content;
    @BindView(R.id.iv_new) ImageView iv_new;
    @BindView(R.id.btn_IUnderstand) Button
btn_IUnderstand;

    public MyViewHolder(View itemView) {
        super(itemView);
        ButterKnife.bind(this, itemView);
    }
}

public MessageAdapter(Map<String, ModelInbox>
messageMap){
    this.messageMap = messageMap;
}

@Override
public MyViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
    View itemview =
LayoutInflater.from(parent.getContext()).
            inflate(R.layout.card_message, parent,
false);
    return new MessageAdapter.MyViewHolder(itemview);
}

@Override
public void onBindViewHolder(final MyViewHolder holder,
int position) {
    final List<String> keySet = new
ArrayList<>(messageMap.keySet());
    final List<ModelInbox> inboxList = new
ArrayList<>(messageMap.values());
    final String key = keySet.get(position);
    final ModelInbox modelInbox =
inboxList.get(position);
    boolean isRead = modelInbox.isRead();
    if(isRead==true){
        holder.iv_new.setVisibility(View.GONE);

holder.btn_IUnderstand.setVisibility(View.GONE);
    }
    else{
```

```
        holder.iv_new.setVisibility(View.VISIBLE);

holder.btn_IUnderstand.setVisibility(View.VISIBLE);
    }

    String response= modelInbox.getResponse();
    if(response.equals("accept")){
        String _content="Your request delivering to "+
modelInbox.getCustomer() + " has been granted.";
        String content = "<font
color='#008000'>"+_content+"</font>";

holder.tv_content.setText(Html.fromHtml(content));
    }
    else if(response.equals("decline")){
        String _content="Your request delivering to "+
modelInbox.getCustomer() + " was rejected. You must
finish the delivery";
        String content = "<font
color='#FF0000'>"+_content+"</font>";

holder.tv_content.setText(Html.fromHtml(content));
    }
    else if(response.equals("abort")){
        String _content="Abort your delivery to " +
modelInbox.getCustomer() + ". The transaction has been
canceled.";
        String content = "<font
color='#5899e2'>"+_content+"</font>";

holder.tv_content.setText(Html.fromHtml(content));
    }


holder.tv_sender.setText(modelInbox.getCompanyName());
    holder.tv_type.setText(modelInbox.getType());

holder.iv_profil.setImageBitmap(modelInbox.getProfilPic
ture());

    holder.btn_IUnderstand.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            new
FireDataMessage().setAsRead(HomeActivity.companyID,
HomeActivity.fleetUID, key, new
DatabaseReference.CompletionListener() {
                @Override
                public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
```

```
holder.iv_new.setVisibility(View.GONE);

holder.btn_IUnderstand.setVisibility(View.GONE);
                    }
                });
            }
        });
}

@Override
public int getItemCount() {
    return messageMap.size();
}
```

Gambar 5.135 US=AS3-3: messageAdapter

.

Gambar 5.136 US-AS3-3: card_message

Terkhusus untuk penugasan yang telah dihapus karena transaksi dibatalkan rentan pada stabilitas aplikasi. Akan tidak ada masalah apabila sopir sedang tidak mengantarkan tugas yang dihapus karena aplikasi tidak sedang menggunakan ranting terkait. Namun apabila sedang menggunakan ranting terkait, dapat membuat aplikasi error. Untuk itu dilakukan 3 pengecekan dalam *activity Task_Ongoing*, yakni:

1. Cek apakah pada sopir bekerja
   Mengecek secara simultan apakah pada ranting *Fleet→FleetID→*working bernilai *true* atau *false*. Apabila bernilai *false* yang artinya penugasan sudah berakhir (*departureID* tidak memiliki nilai), maka langsung masuk ke *activity* utama *HomeActivity*. Kode yang digunakan adalah pada gambar 5.137.

2. Cek apakah *task* yang sedang diantarkan dihapus
   Mengecek secara simultan apakah pada ranting *departure→departureID→task→taskID* masih memiliki nilai. Ranting tersebut tidak bernilai saat perusaahaan menghapus *task* tersebut. Apabila tidak bernilai, maka aplikasi menghapus ranting *departure-→departureID→ task*Working dan mengganti nilai ranting *departure→departureID→isWorking* menjadi *false*. Kemudian mengembalikan tampilan layar menjadi kondisi memilih tugas yang ingin diantarkan lalu mengambil data tugas apa yang harus diantarkan. Kode yang digunakan adalah pada gambar 5.138.

3. Cek apakah semua *task* pada *DepartureID* sudah selesai
   Mengecek apakah semua nilai pada ranting *departure→ departureID→task→taskID* sudah selesai, ada yang belum selesai, atau bahkan dihapus semua oleh perusahaan. Apabila pada ranting *task* tersebut tidak ada nilai, maka aplikasi langsung menjalankan *activity HomeActivity*. Begitu jug ajika semua *taskID* sudah selesai, langsung menjalankan *activity HomeActivity*. Kode yang digunakan adalah pada gambar 5.139.

```
workEventListener = new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot
dataSnapshot, String s) {

if(dataSnapshot.getKey().equals(HomeActivity.fleetU
ID)){

if(dataSnapshot.child(FireDataUser.WORKING).getValu
e().toString().equals("false")){
                startActivity(new
Intent(getApplicationContext(),
HomeActivity.class));
                finish();
            }
        }
    }

    @Override
    public void onChildChanged(DataSnapshot
dataSnapshot, String s) {

if(dataSnapshot.getKey().equals(HomeActivity.fleetU
ID)){

if(dataSnapshot.child(FireDataUser.WORKING).getValu
e().toString().equals("false")){
                startActivity(new
Intent(getApplicationContext(),
HomeActivity.class));                finish();
            }
        }
    }

    @Override
    public void onChildRemoved(DataSnapshot
dataSnapshot) {    }

    @Override
    public void onChildMoved(DataSnapshot
dataSnapshot, String s) {    }

    @Override
    public void onCancelled(DatabaseError
databaseError) {    }
};
new
FireDataUser().ref.addChildEventListener(workEventL
istener);
```

Gambar 5.137 US-AS3-3: Cek apakah sopir bekerja

```
public void cancel(){
    new
FireDataDeparture().writeCancelWorking(depart
ureID, new
DatabaseReference.CompletionListener() {
        @Override
        public void onComplete(DatabaseError
databaseError, DatabaseReference
databaseReference) {
            if(databaseError==null){
                taskID = null;
                wareHouseName = null;
                custLatLng = null;
                wareHouseLatLng = null;
                mMap.clear();
                setMarker();

                loadtask();
                idlingView();
            }
        }
    });
}

abortEventListener = new ChildEventListener()
{
    @Override
    public void onChildAdded(DataSnapshot
dataSnapshot, String s) {

if(dataSnapshot.getKey().toString().equals(de
partureID)){
            if(taskID != null){

if(dataSnapshot.child("task").child(taskID).g
etValue() == null){
                    cancel();
                }
            }
        }
        checkDepartureComplete();
    }

    @Override
    public void onChildChanged(DataSnapshot
dataSnapshot, String s) {
```

```
if(dataSnapshot.getKey().toString().equals(de
partureID)){
            if(taskID != null){

if(dataSnapshot.child("task").child(taskID).g
etValue() == null){
                    cancel();
                }
            }
        }
        checkDepartureComplete();
    }

    @Override
    public void onChildRemoved(DataSnapshot
dataSnapshot) {

    }

    @Override
    public void onChildMoved(DataSnapshot
dataSnapshot, String s) {

    }

    @Override
    public void onCancelled(DatabaseError
databaseError) {

    }
};
new
FireDataDeparture().ref.addChildEventListener
(abortEventListener);
```

Gambar 5.138 US-AS3-3: Cek apakah task yang sedang diantarkan dihapus

```
 private void checkDepartureComplete(){
    departureEventListener2 = new
ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {

if(dataSnapshot.getValue()==null){
                new
FireDataDeparture().writeCompletedDeparture(d
epartureID, new
DatabaseReference.CompletionListener() {
                    @Override
                    public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                        new
FireDataUser().writeUserStopWorking(HomeActiv
ity.fleetUID, new
DatabaseReference.CompletionListener() {
                            @Override
                            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                                new
FireDataUser().ref.child(HomeActivity.fleetUI
D).child(FireDataUser.DEPARTUREID).removeValu
e();

startActivity(new
Intent(getApplicationContext(),
HomeActivity.class));
                                finish();
                            }
                        });
                    }
                });
            }
            else{
                String isTaskComplete=null;
                for(DataSnapshot postSnapshot
: dataSnapshot.getChildren()){
                    isTaskComplete =
postSnapshot.child("isTaskCompleted").getValu
e().toString();
                    Log.d("tsss", "task
```

```
completion status: "+ isTaskComplete);

if(isTaskComplete.equals("false")){
                        break;
                    }
                }

                // If all task is complete
                if
(!isTaskComplete.equals("false")) {
                    Log.d("tsss", "all
delivery is complete");
                    new
FireDataDeparture().writeCompletedDeparture(d
epartureID, new
DatabaseReference.CompletionListener() {
                        @Override
                        public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {

                            new
FireDataUser().writeUserStopWorking(HomeActiv
ity.fleetUID, new
DatabaseReference.CompletionListener() {
                                @Override
                                public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                                    new
FireDataUser().ref.child(HomeActivity.fleetUI
D).child(FireDataUser.DEPARTUREID).removeValu
e();

startActivity(new
Intent(getApplicationContext(),
HomeActivity.class));
                                    finish();
                                }
                            });
                        }
                    });
                }
            }
        }
        @Override
```

```
        public void onCancelled(DatabaseError
databaseError) {
        }
    };
    new
FireDataDeparture().ref.child(departureID).ch
ild("task").addValueEventListener(departureEv
entListener2);
}
```

Gambar 5.139 US-AS3-3: Cek apakah semua task pada departureID sudah selesai

5.3.2.10 US-AS4-1: Mengirim lokasi ke *server* secara simultan

Pada *HomeActivity* ada tombol *reCenter* untuk memusatkan peta pada lokasi armada saat ini, dan memperbaharui *marker*-nya pada peta. Kode yang digunakan adalah pada gambar 5.140.

Ada juga Fungsi yang dijalankan pada *HomeActivity*. Saat ada perubahan minimal 2 detik atau minimal jarak 20m, lokasi akan diperbaharui dengan mengggunakan *LocationList*ener. Berikut adalah kode yang digunakan adalah pada gambar 5.141.

Selain pada *HomeActivity*, Aktivitas *TaskOngoing* juga memiliki *LocationListener* yang bisa digunakan untuk mengambil lokasi sopir terkini lalu menyimpannya dalam database. Kode yang digunakan adalah pada gambar 5.142.

```
@OnClick(R.id.iv_recenter) public void recenterMap(){
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }
    Location location =
LocationServices.FusedLocationApi.getLastLocation(googl
eApiClient);
    if(location != null){
        fleetLoc = new LatLng(location.getLatitude(),
location.getLongitude());
        new FireDataUser().writeUserLocation(fleetUID,
fleetLoc, new DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                Log.d("tsss", "Location Update:
"+fleetLoc);
                if (fleetMarker != null){
                    fleetMarker.remove();
                }
                fleetMarker = mMap.addMarker(new
MarkerOptions().position(fleetLoc).title("Fleet").

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_fleet_current_location", 59, 100))));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(fleet
Loc, 17));
            }
        });
    }
}
```

Gambar 5.140 US-AS4-1: memusatkan kembali lokasi pada peta

```
LocationListener mLocationListener = new
LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        fleetLoc = new LatLng(location.getLatitude(),
location.getLongitude());
        new FireDataUser().writeUserLocation(fleetUID,
fleetLoc, new DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                if(databaseError==null){
                    Log.d("tsss", "Location Update:
"+fleetLoc);
                    if (fleetMarker != null){
                        fleetMarker.remove();
                    }
                    fleetMarker = mMap.addMarker(new
MarkerOptions().position(fleetLoc).title("Fleet").

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap("m
ap_fleet_current_location", 59, 100))));
                }
            }
        });
    }

    @Override
    public void onStatusChanged(String provider, int
status, Bundle extras) {    }

    @Override
    public void onProviderEnabled(String provider) {  }

    @Override
    public void onProviderDisabled(String provider) { }
};

LocationManager mLocationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED
        && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {

mLocationManager.requestLocationUpdates(LocationManager
.GPS_PROVIDER, 2000,20, mLocationListener);
}
```

Gambar 5.141 US-AS4-1: HomeActivity LocationListener

```
private void getCurrentLocation() {
    LocationListener mLocationListener = new
LocationListener() {
        @Override
        public void onLocationChanged(Location
location) {
            fleetLatLng = new
LatLng(location.getLatitude(),
location.getLongitude());
            new
FireDataUser().writeUserLocation(HomeActivity.fleetUID,
fleetLatLng, new DatabaseReference.CompletionListener()
{
                @Override
                public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {

                }
            });

            //Task location and Fleet marker
        }

        @Override
        public void onStatusChanged(String provider,
int status, Bundle extras) {        }

        @Override
        public void onProviderEnabled(String provider)
{        }

        @Override
        public void onProviderDisabled(String provider)
{        }
    };

    LocationManager mLocationManager =
(LocationManager) getSystemService(LOCATION_SERVICE);
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED
            && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {

mLocationManager.requestLocationUpdates(LocationManager
.GPS_PROVIDER, 2000,20, mLocationListener);
    }
}
```

Gambar 5.142 US-AS4-1: TaskOngoing LocationListener

5.3.2.11 US-AS4-2: Melihat lokasinya terkini, lokasi gudang, dan lokasi perusahaan

Pada *HomeActivity*, terdapat *fragment* MapActivity sehingga *user* bisa melihat lokasi perusahaan, lokasinya saat ini, dan lokasi gudang-gudang dalam visualisasi peta. Yang pertama adalah mengambil lokasi terkini *user* menggunakan kode seperti pada gambar 5.143. Kemudian untuk setiap perubahan lokasi kendaraan, aplikasi menggunakan *LocationListener* dengan kode pada gambar 5.141.

Setelah itu aplikasi mengambil data lokasi perusahaan dan gudang-gudang kemudian meletakkan *marker* di peta sesuai yang ada di *Firebase*. Untuk mengambil data gudang digunakan kode pada gambar 5.144, dan untuk mengambil alamat perusahaan digunakan kode pada gambar 5.145. Data gudang diambil menggunakan *ChildEventListener* karena lokasi ada kemungkinan gudang bertambah atau berkurang atau pindah alamat. Alamat perusahaan menggunakan *ValueEventListener* karena alamat perusahaan sangat jarang untuk berpindah.

5.3.2.12 US-AS5-1: Mengedit informasi akun

Detail informasi akun ditampilkan dalam Aktivitas *HomeActivity Fragment AccountActivity* dengan tampilan pada gambar 5.146. Informasi akun yang ditampilkan adalah nama lengkap, *email*, *password*, lisensi SIM *user*, nomor telepon, alamat, jenis kelamin, tanggal lahir, nama truk, dan nomor plat truk yang dikendarai.

```
private void getFleetLocation(){
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&

ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }
    Location location =
LocationServices.FusedLocationApi.getLastLocation
(googleApiClient);
    if(location != null){
        fleetLoc = new
LatLng(location.getLatitude(),
location.getLongitude());
        new
FireDataUser().writeUserLocation(fleetUID,
fleetLoc, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference
databaseReference) {
                Log.d("tsss", "Location Update:
"+fleetLoc);
                if (fleetMarker != null){
                    fleetMarker.remove();
                }
                fleetMarker = mMap.addMarker(new
MarkerOptions().position(fleetLoc).title("Fleet")
.

icon(BitmapDescriptorFactory.fromBitmap(resizeBit
map("map_fleet_current_location", 59, 100))));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom
(fleetLoc, 17));
            }
        });
    }
}
```

Gambar 5.143 US-AS4-1: Mengambil data lokasi user

```
private void getWareHouseData(){
    wareHouseEventListener = new
ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot
dataSnapshot, String s) {
            ModelWarehouse modelWarehouse =
dataSnapshot.getValue(ModelWarehouse.class);

if(companyID.equals(modelWarehouse.getCompanyID())){
                ModelAddress addressWareHouse =
dataSnapshot.child("address").getValue(ModelAddress.
class);
                LatLng _temp = new
LatLng(addressWareHouse.getLatitude(),
addressWareHouse.getLongitude());
                wareHouseList.add(_temp);
                for(int i=0;
i<wareHouseList.size();i++){
                    wareHouseMarker =
mMap.addMarker(new
MarkerOptions().position(_temp).title(modelWarehouse
.getName()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap
("map_warehouse", 52, 100))));
                }
            }
        }

        @Override
        public void onChildChanged(DataSnapshot
dataSnapshot, String s) {
            ModelWarehouse modelWarehouse =
dataSnapshot.getValue(ModelWarehouse.class);

if(companyID.equals(modelWarehouse.getCompanyID())){
                ModelAddress addressWareHouse =
dataSnapshot.child("address").getValue(ModelAddress.
class);
                LatLng _temp = new
LatLng(addressWareHouse.getLatitude(),
addressWareHouse.getLongitude());
                if(!wareHouseList.contains(_temp)){
                    wareHouseList.add(_temp);
                    for(int i=0;
i<wareHouseList.size();i++){
                        wareHouseMarker =
```

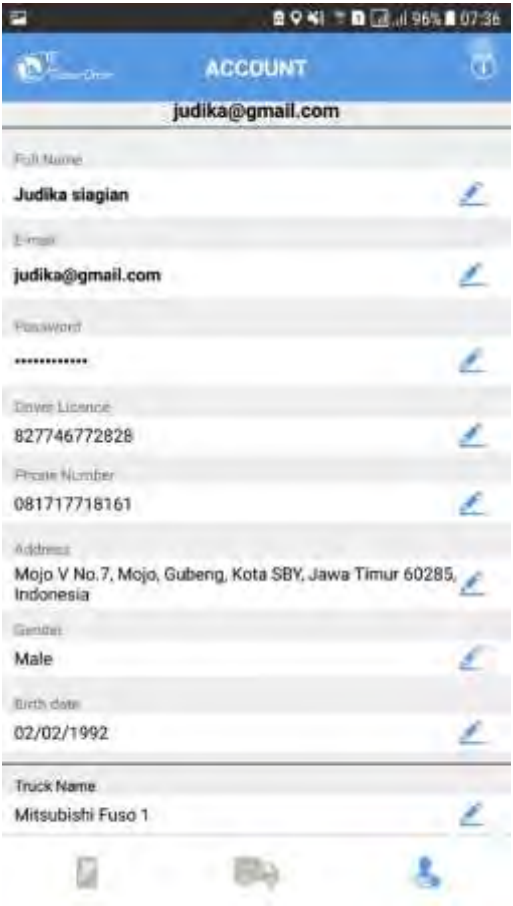
```
mMap.addMarker(new
MarkerOptions().position(_temp).title(modelWarehouse
.getName()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap
("map_warehouse", 52, 100))));
                    }
                }
            }
        }

        @Override
        public void onChildRemoved(DataSnapshot
dataSnapshot) {
            ModelWarehouse modelWarehouse =
dataSnapshot.getValue(ModelWarehouse.class);

if(companyID.equals(modelWarehouse.getCompanyID())){
                ModelAddress addressWareHouse =
dataSnapshot.child("address").getValue(ModelAddress.
class);
                LatLng _temp = new
LatLng(addressWareHouse.getLatitude(),
addressWareHouse.getLongitude());
                if(!wareHouseList.contains(_temp)){
                    wareHouseList.remove(_temp);
                    for(int i=0;
i<wareHouseList.size();i++){
                        wareHouseMarker =
mMap.addMarker(new
MarkerOptions().position(_temp).title(modelWarehouse
.getName()).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap
("map_warehouse", 52, 100))));
                    }
                }
            }
        }

        @Override
        public void onChildMoved(DataSnapshot
dataSnapshot, String s) {        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {
```

```
        }
    };
    new
FireDataWareHouse().ref.addChildEventListener(wareHo
useEventListener);
}
```

Gambar 5.144 US-AS4-2: Mengambil data lokasi gudang

```
private void getCompanyData(){
    companyEventListener = new ValueEventListener()
{
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {
            companyToken =
dataSnapshot.child("token").getValue().toString();
            ModelAddress address =
dataSnapshot.child("address").getValue(ModelAddress.
class);
            companyLoc = new
LatLng(address.getLatitude(),
address.getLongitude());
            Log.d("tsss", "Company Coordinate: " +
companyLoc);

            if (companyMarker != null){
                companyMarker.remove();
            }
            companyMarker = mMap.addMarker(new
MarkerOptions().position(companyLoc).title("Company"
).

icon(BitmapDescriptorFactory.fromBitmap(resizeBitmap
("map_company", 52, 100))));
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }
    };
    new
FireDataCompany().ref.child(companyID).addListenerFo
rSingleValueEvent(companyEventListener);
}
```

Gambar 5.145 US-AS4-2: Mengambil data lokasi perusahan



Gambar 5.146 US-AM5-1: Tampilan AccountAcitvity

Untuk mengedit masing-masing data, maka *user* harus menekan ikon pensil sehingga bisa masuk ke *activity* untuk mengedit informasi akun. Berikut adalah potongan kode untuk mengedit masing-masing data:

1. Nama lengkap

Untuk mengedit nama lengkap *user*, maka aplikasi mengambil teks yang ada dalam *EditText* lalu menyimpannya dalam *database* menggunakan kode pada gambar 5.147.

```
@OnClick(R.id.btn_save_fullname) public void
btn_save_fullname(){
    newFullName =
et_edit_fullname.getText().toString();

    if(TextUtils.isEmpty(newFullName)){
        Toast.makeText(this, "new full name is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUser().writeUserFullName(user.getUid(),
newFullName, new
DatabaseReference.CompletionListener() {
            @Override
            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(EditFullName.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.147 US-AS5-1: Edit nama lengkap sopir

2. *Email*

   Karena *email* merupakan cara *user* untuk *login*/otentikasi, maka jika ingin merubah *email*, aplikasi harus merubahnya

pada *Firebase database* dan *Firebase authentication*. Untuk memperbaharuinya, pastikan *user* pada sesi yang sama dengan ketika *login*. Jika aplikasi sudah pernah ditutup tanpa *log out*, *edit email* tidak bisa berfungsi. Itu adalah kebijakan dari *Firebase* sendiri. Kode dibawah ini digunakan untuk mengedit *email user*. Diawali dengan memperbaharui *email* di bagian otentikasi, baru dilanjutkan pada bagian *database*. Sebelum menyimpan, aplikasi juga mengecek apakah email yang dimasukkan sudah sesuai dengan format email atau tidak. Kode yang digunakan adalah pada gambar 5.148.

3. *Password*

   Password adalah hal sensitif yang tidak boleh diketahui orang lain. Oleh karena itu untuk menggantinya dimulai dari *user* mengisi *password* lamanya, kemudian *password* baru diisikan 2 kali. Aplikasi akan mengambil data *password* lama dari *Firebase* untuk mencocokkan apakah data yang diisikan sudah sesuai atau belum. Jika sudah sesuai aplikasi mencocokkan apakah 2 *password* baru yang diisikan sudah sama persis. Kode yang digunakan adalah pada gambar 5.149. Sama seperti mengedit *email user*, *password* harus diedit pada bagian authentication dan *database*, dan pada sesi yang sama dengan *login*. Kode yang digunakan adalah pada gambar 5.150.

4. Lisensi SIM *user*

   Untuk mengedit lisensi *user*, maka aplikasi mengambil teks yang ada dalam *EditText* lalu menyimpannya dalam *database*. Tidak ada aturan dan batasan yang didefinisikan untuk mengisi nomor SIM *user*. Kode yang digunakan adalah pada gambar 5.151.

```
@OnClick(R.id.btn_save_email) public void
btn_save_email(){
    newEmail = et_edit_email.getText().toString();

    String expression = "^[\\w\\.-]+@([\\w\\-]+\\.)+[A-
Z]{2,4}$";
    Pattern pattern = Pattern.compile(expression,
Pattern.CASE_INSENSITIVE);
    Matcher matcher = pattern.matcher(newEmail);

    if(matcher.matches()){
        if(TextUtils.isEmpty(newEmail)){
            Toast.makeText(this, "new email is empty",
Toast.LENGTH_SHORT).show();
            return;
        }
        else{

user.updateEmail(newEmail).addOnCompleteListener(new
OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void>
task) {
                    if(task.isSuccessful()){
                        new
FireDataUser().writeUserEmail(user.getUid(), newEmail, new
DatabaseReference.CompletionListener() {
                            @Override
                            public void
onComplete(DatabaseError databaseError, DatabaseReference
databaseReference) {
                                if(databaseError == null){
                                    finish();
                                }
                                else{

Toast.makeText(EditEmail.this, "Error while updating : " +
databaseError.getDetails(), Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                    }
                    else{
                        Toast.makeText(EditEmail.this,
"Error updating. Please log out then log in again",
Toast.LENGTH_SHORT).show();
                    }
                }
            }).addOnFailureListener(new
OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception
```

```
e) {

Toast.makeText(getApplicationContext(), "error: " +
e.getMessage(), Toast.LENGTH_LONG).show();
                    return;
                }
            });
        }
    }
    else{
        Toast.makeText(getApplicationContext(), "input
format invalid: ", Toast.LENGTH_LONG).show();
    }
}
```

Gambar 5.148 US-AS5-1: Edit email sopir

```
@OnClick(R.id.btn_save_password) public void
btn_save_password() {
    password1 = et_new_password1.getText().toString();
    password2 = et_new_password2.getText().toString();
    oldPassword = et_old_password.getText().toString();

    if(TextUtils.isEmpty(password1)){
        Toast.makeText(this, "new password is empty",
Toast.LENGTH_SHORT).show();
        return;
    }
    else if(TextUtils.isEmpty(password2)){
        Toast.makeText(this, "retype new password",
Toast.LENGTH_SHORT).show();
        return;
    }
    else if(TextUtils.isEmpty(oldPassword)){
        Toast.makeText(this, "fill the old password",
Toast.LENGTH_SHORT).show();
        return;
    }
    else if(!password1.equals(password2)){
        Toast.makeText(this, "new password didn't match",
Toast.LENGTH_SHORT).show();
        return;
    }
    else if(!savedPassword.equals(oldPassword)){
        Toast.makeText(this, "old password didn't match",
Toast.LENGTH_SHORT).show();
        return;
    }
}
```

Gambar 5.149 US-AS5-1: Cek kecocokan password lama, cek ketepatan password baru

```
user.updatePassword(password1).addOnCompleteL
istener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull
Task<Void> task) {
        new
FireDataUser().writeUserPassword(user.getUid(
), password1, new
DatabaseReference.CompletionListener() {
            @Override
            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(EditPassword.this, "Error
while updating : " +
databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}).addOnFailureListener(new
OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception
e) {

Toast.makeText(getApplicationContext(),
"error: " + e.getMessage(),
Toast.LENGTH_LONG).show();
        return;
    }
});
```

Gambar 5.150 US-AS5-1: Edit password sopir

```
@OnClick(R.id.btn_save_license) public void
btn_save_license(){
    newLicense =
et_edit_license.getText().toString();

    if(TextUtils.isEmpty(newLicense)){
        Toast.makeText(this, "new license is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUser().writeUserLicense(user.getUid(),
newLicense, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(EditLicense.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.151 US-AS5-1: Edit lisensi sim sopir

5. Nomor telepon

   Untuk mengedit nomor telepon, maka aplikasi mengambil teks yang ada dalam *EditText* lalu menyimpannya dalam *database*. Tidak ada aturan dan batasan yang didefinisikan untuk mengisi nomor telepon. Kode yang digunakan adalah pada gambar 5.152

```
@OnClick(R.id.btn_save_phonenumber) public
void btn_save_phonenumber(){
    newPhone=
et_edit_phonenumber.getText().toString();

    if(TextUtils.isEmpty(newPhone)){
        Toast.makeText(this, "new phone
number is empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUser().writeUserPhone(user.getUid(),
newPhone, new
DatabaseReference.CompletionListener() {
            @Override
            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(EditPhone.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.152 US-AS5-1: Edit nomor telepon sopir

6. Alamat

   Untuk mengedit alamat, *user* harus mengambil kordinat alamat rumahnya pada peta menggunakan *activity* **Account_Set_Map**. Setelah didapatkan, lalu menyimpannya ke dalam *database* menggunakan kode pada gambar 5.153.

```
@Override protected void onActivityResult(int
requestCode, int resultCode, Intent data){
    if(requestCode==3){
        if(resultCode == Activity.RESULT_OK){
            latitude =
Double.parseDouble(data.getStringExtra("latitude"))
;
            longitude =
Double.parseDouble(data.getStringExtra("longitude")
);
            latLng = new LatLng(latitude,
longitude);

            Geocoder geocoder;
            List<Address> addresses;
            geocoder = new Geocoder(this,
Locale.getDefault());
            try {
                addresses =
geocoder.getFromLocation(latitude, longitude,1);
                address =
addresses.get(0).getAddressLine(0);
                et_edit_address.setText(address);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

@OnClick(R.id.tv_setMap) public void setMap(){
    Intent intent = new Intent(EditAddress.this,
Account_Set_Map.class);
    startActivityForResult(intent, 3);
}

@OnClick(R.id.btn_save_address) public void
btn_save_address(){
    address = et_edit_address.getText().toString();
    if(latLng==null){
        Toast.makeText(this, "Set address with a
map", Toast.LENGTH_SHORT).show();
        return;
    }
    if(TextUtils.isEmpty(address)){
        Toast.makeText(this, "new address is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUser().writeUserAddress(user.getUid(),
```

```
latLng, new DatabaseReference.CompletionListener()
{
            @Override
            public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference)
{
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(EditAddress.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.153 US-AS5-1: Edit alamat sopir

7. Jenis kelamin
   Untuk mengedit jenis kelamin, maka aplikasi mengambil teks yang ada dalam *EditText* (male/female) lalu menyimpannya dalam *database*. Namun tidak ada aturan dan batasan yang didefinisikan agar *user* tidak mengisi nilai selain itu. Kode yang digunakan adalah pada gambar 5.154.
8. Tanggal lahir
   Untuk mengedit tanggal lahir, maka aplikasi mengambil teks yang ada dalam *EditText* (MM/DD/YYYY). Namun tidak ada aturan dan batasan yang didefinisikan agar *user* tidak mengisi data selain dari format itu. Kode yang digunakan adalah pada gambar 5.155.
9. Nama truk
   Untuk mengedit nama truk, maka aplikasi mengambil teks yang ada dalam *EditText*, lalu menyimpannya dalam *database* menggunakan kode pada gambar 5.156

```
@OnClick(R.id.btn_save_gender) public void
btn_save_gender(){
    newGender = et_edit_gender.getText().toString();

    String gender = newGender.trim().toLowerCase();
    if(gender.equals("male") || gender.equals("female")) {
        if(TextUtils.isEmpty(gender)){
            Toast.makeText(this, "new gender is empty",
Toast.LENGTH_SHORT).show();
            return;
        }
        else{
            new
FireDataUser().writeUserGender(user.getUid(), gender, new
DatabaseReference.CompletionListener() {
                @Override
                public void onComplete(DatabaseError
databaseError, DatabaseReference databaseReference) {
                    if(databaseError == null){
                        finish();
                    }
                    else{
                        Toast.makeText(EditGender.this,
"Error while updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    }
    else{
        Toast.makeText(this, "input format invalid",
Toast.LENGTH_SHORT).show();
    }
}
```

Gambar 5.154 US-AS5-1: Edit jenis kelamin sopir

10. Nomor plat truk

    Untuk mengedit nomor plat truk, maka aplikasi mengambil teks yang ada dalam *EditText*. Namun tidak ada aturan dan batasan yang didefinisikan agar *user* tidak mengisi data selain format penomoran plat yang resmi. Kode yang digunakan adalah pada gambar 5.157.

```
private boolean validate(String date){
    String expression = "(0?[1-9]|1[012])+[/.-
```

```
]+(0?[1-9]|[12][0-9]|3[01])+[/.-]+((19|20)\\d\\d)";
    Matcher matcher =
Pattern.compile(expression).matcher(date);

    if(matcher.matches()){
        String value = matcher.group(0);
        int month =
Integer.parseInt(value.substring(0,2));
        int day =
Integer.parseInt(value.substring(3,5));
        int year =
Integer.parseInt(value.substring(6));
        Log.d("tsss","Value: "+ month + " : " + day
+ " : " + year);

        if ((day<0 || day>30) && (month==4 ||
month==6 || month==9 ||month==11)) {

            Log.d("tsss", "birthdate check: false
month 4,6,9,11");
            return false;
        }
        else if((day<0 || day>31) && (month==1 ||
month==3 || month==5 ||month==7 ||month==8
||month==10 ||month==12)){
            Log.d("tsss", "birthdate check: false
month 1,3,5,7,8,10,12");
            return false;
        }
        else if (month==2) {
            if(year % 4==0){
                if(day<0 || day>29){
                    Log.d("tsss", "birthdate check:
false month 2 date 30,31");
                    return false;
                }
                else{
                    return true;
                }
            }
            else{
                if(day>28){
                    Log.d("tsss", "birthdate check:
false month 2 date 29,30,31");
                    return false;
                }
                else{
                    return true;
                }
            }
        }
        else if(month<0 || month >12){
```

```
            return false;
        }
        else{
            return true;
        }
    }
    else{
        Log.d("tsss", "matcher no match");
        return false;
    }
}

@OnClick(R.id.btn_save_birthday) public void
btn_save_birthday(){
    newBirthDate =
et_edit_birthdate.getText().toString();

    if(TextUtils.isEmpty(newBirthDate)){
        Toast.makeText(this, "new birthdate is
empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else if(newBirthDate.length()!=10){
        Toast.makeText(EditBirthdate.this, "input
format invalid", Toast.LENGTH_SHORT).show();
    }
    else{
        boolean check = validate(newBirthDate);
        if(check){
            new
FireDataUser().writeUserBirthDate(user.getUid(),
newBirthDate, new
DatabaseReference.CompletionListener() {
                @Override
                public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                    if(databaseError == null){
                        finish();
                    }
                    else{

Toast.makeText(EditBirthdate.this, "Error while
updating : " + databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
        else{
            Toast.makeText(EditBirthdate.this,
"input format invalid", Toast.LENGTH_SHORT).show();
```

```
        }
    }
}
```

Gambar 5.155 US-AS5-1: Edit tanggal lahir sopir

```
@OnClick(R.id.btn_save_vehiclename) public
void btn_save_vehiclename(){
    newTruckName =
et_edit_vehiclename.getText().toString();

    if(TextUtils.isEmpty(newTruckName)){
        Toast.makeText(this, "new truck name
is empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUser().writeUserTruckName(user.getUid
(), newTruckName, new
DatabaseReference.CompletionListener() {
            @Override
            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(EditTruckName.this, "Error
while updating : " +
databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.156 US-AS5-1: Edit nama truk sopir

```
@OnClick(R.id.btn_save_licenseplate) public
void btn_save_licenseplate(){
    newPlate =
et_edit_licenseplate.getText().toString();

    if(TextUtils.isEmpty(newPlate)){
        Toast.makeText(this, "new Vehicle
Plate is empty", Toast.LENGTH_SHORT).show();
        return;
    }
    else{
        new
FireDataUser().writeUserTruckPlate(user.getUi
d(), newPlate, new
DatabaseReference.CompletionListener() {
            @Override
            public void
onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
                if(databaseError == null){
                    finish();
                }
                else{

Toast.makeText(EditTruckPlate.this, "Error
while updating : " +
databaseError.getDetails(),
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Gambar 5.157 US-AS5-1: Edit nomor plat truk sopir

5.3.2.13 US-AS6-1: Reset Password

Fitur reset password ada terdapat dalam Aktivitas *SignInActivity seperti pada gambar 5.99*. Saat user lupa atas Password miliknya, user bisa mengajukan reset password yang nantinya akan dikelola langsung dari Firebase. Untuk bisa

melakukannya, user harus tahu email yang ia gunakan, kemudian Firebase akan mengirimkan email untuk mengubah password. Kode yang digunakan adalah seperti gambar 5.159.

```
@OnClick(R.id.reset_hint) public void reset(){
    email = et_id.getText().toString();
    if(TextUtils.isEmpty(email)){
        Toast.makeText(this, "Please enter your
email without password",
Toast.LENGTH_SHORT).show();
        return;
    }
    else{

mAuth.sendPasswordResetEmail(email).addOnComplete
Listener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull
Task<Void> task) {

Toast.makeText(getApplicationContext(), "Check
your email", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Gambar 5.158 US-AS6-1: Reset Password

5.4 *Desktop* Manajemen Perusahaan

5.4.1 Lingkungan Aplikasi

Tabel 5.3 Lingkungan aplikasi fleemo website

Framework	Bootstrap v4
Bahasa Pemrograman	Hypertext Preprocessing v5
Font	Helvetica Neue
Database	Firebase Database

Pada bagian ini dijelaskan proses implementasi setiap fungsi aplikasi dengan potongan kode yang digunakan. IDE yang dipakai adalah **Pinegrow Web Editor 4**. Berikut adalah fungsi-fungsi yang dimiliki oleh *Fleemo Driver*, sesuai dengan User Story pada Tabel 4-5.

5.4.1.1 US-WM1-1 Login Website

Login website hanya bisa menggunakan email perusahaan, sementara email armada dilarang. Untuk itu setelah user mengisikan email dan passwordnya, aplikasi menjalankan fungsi checkAccount yang mengambil data-data email yang terdaftar pada ranting ***user*** pada Firebase. Jika ada, maka akan menjalankan fungsi sign in ke Firebase, apabila tidak ada, maka akan memunculkan pesan email salah. Tampilan website saat sign in adalah seperti gambar 5.159. Kemudian Kode yang digunakan untuk Sign in adalah seperti gambar 5.160.

Gambar 5.159 US-WM1-1: Tampilan login website

```
function onClick() {
  $('#login-btn').click(function() {
    var _email = $('#login-email').val();
    var _password = $('#login-password').val();

    checkAccount(_email, _password);
  });
}

function checkAccount(email, password) {
  refUsers.once("value", function(snap) {
    var countUsers = snap.numChildren();
    var countLoop = 0;

    snap.forEach(function(childSnapshot) {
      countLoop++;
      // var _key = childSnapshot.key;
      var _email = childSnapshot.child("email").val();

      if (email == _email) {
        infoValidation(true);
        loginAccount(email, password);
        return true;
      }
      if        (countUsers        ==        countLoop)
infoValidation(false);
    });
  });
}

function loginAccount(email, password) {
  if (email != "" && password != "") {
    infoValidation(true);
    firebase.auth().signInWithEmailAndPassword(email,
password).catch(function(error) {
      infoValidation(false);
    });
  } else infoValidation(false);
}

function infoValidation(isValid) {
  if (isValid) {
    $('#login-validate').text("");
    $('#login-validate').css("display", "none");
  } else {
    $('#login-validate').text("*wrong      email      or
password");
    $('#login-validate').css("display", "");
  }
}
```

Gambar 5.160 US-WM1-1: fungsi login website

5.4.1.2 US-WM2-1 Melihat Lokasi Seluruh Armada

Tampilan utama dari website adalah sebuah peta yang menampilkan lokasi perusahaan, persebaran lokasi gudang dan armada milik perusahaan yang masing-masingnya ditampilkan dalam bentuk pin yang berbeda-beda. Untuk menampilkan lokasi tersebut, pertama-tama, aplikasi mengambil dulu kordinat-kordinat lokasi dalam firebase, lalu menyimpannya dalam list.

1. Kordinat lokasi perusahaan
   Lokasi perusahaan diambil dari Firebase pada ranting user→userID→address. Kode yang digunakan adalah pada gambar 5.161.

2. Kordinat lokasi armada
   Lokasi armada diambil dari Firebase pada ranting fleet→fleetID→location. Kode yang digunakan adalah pada Gambar 6-162.

3. Kordinat lokasi gudang
   Lokasi gudang diambil dari Firebase pada ranting warehouse→warehouseID→address. Kode yang digunakan adalah pada gambar 5.164.

Setelah data-data tersebut didapatkan barulah peta diinisiasi dan meletakkan kordinat-kordinat tersebut dengan kordinat yang dikustomisasi sesuai dengan jenisnya. Kode yang digunakan adalah pada gambar 5.165.

```
var refUsers = rootRef. child("fleet");

function monitorCompany() {
  refUsers.on('child_added', function(data) {
    var _companyID = data.key;

    if (_companyID == userUID) {
      var _lat =
data.child("address").child("latitude").val();
      var _lng =
data.child("address").child("longitude").val();
      companyName = data.child("fullname").val();
      companyLatLng = {
        lat: _lat,
        lng: _lng
      };

      initMarker();
    }
  });

  refUsers.on('child_changed', function(data) {
    var _companyID = data.key;
    var _name = data.child("name").val();

    if (_companyID == userUID) {
      var _lat =
data.child("address").child("latitude").val();
      var _lng =
data.child("address").child("longitude").val();
      companyName = data.child("fullname").val();
      companyLatLng = {
        lat: _lat,
        lng: _lng
      };

      initMarker();
    }
  });

  refUsers.on('child_removed', function(data) {
    // do nothing
  });
}
```

Gambar 5.161 US-WM2-1: Mengambil kordinat lokasi perusahaan

```
var refWarehouse = rootRef.child("warehouse");

function monitorFleet() {
  var flagArray = 0;
  refFleet.on('child_added', function(data) {
    var _companyID =
data.child("companyID").val();
    var _email = data.child("email").val();
    var _lat =
data.child("location").child("latitude").val();
    var _lng =
data.child("location").child("longitude").val();
    var _fullname = data.child("fullname").val();
    var _working = data.child("working").val();

    if (_companyID == userUID) {
      if (_working == false) {
        arrEmailFleet1.push(_email);
        arrLocFleet1[_email] = [_fullname, _lat,
_lng];
      } else {
        arrEmailFleet2.push(_email);
        arrLocFleet2[_email] = [_fullname, _lat,
_lng];
      }

      initMarker();
    }

    flagArray++;
  });

  refFleet.on('child_changed', function(data) {
    var _companyID =
data.child("companyID").val();
    var _email = data.child("email").val();
    var _lat =
data.child("location").child("latitude").val();
    var _lng =
data.child("location").child("longitude").val();
    var _fullname = data.child("fullname").val();
    var _working = data.child("working").val();

    if (_companyID == userUID) {
      if (_working == false) {
        arrLocFleet1[_email] = [_fullname, _lat,
_lng];
        initMarker();
      } else {
        arrLocFleet2[_email] = [_fullname, _lat,
_lng];
        initMarker();
```

```
      }

    }
  });
}
```

Gambar 5.162 US-WM2-1: Mengambil kordinat lokasi armada

```
var refWarehouse = rootRef.child("warehouse");

function monitorWarehouse() {
var flagArray = 0;
refWarehouse.on('child_added', function(data) {
var _companyID = data.child("companyID").val();
var _id = data.key;
var _lat =
data.child("address").child("latitude").val();
var _lng =
data.child("address").child("longitude").val();
var _name = data.child("name").val();

if (_companyID == userUID) {
arrIDWarehouse.push(_id);
arrLocWarehouse[_id] = [_name, _lat, _lng];
initMarker();
}

flagArray++;
});

refWarehouse.on('child_changed', function(data) {
var _companyID = data.child("companyID").val();
var _id = data.key;
var _lat =
data.child("address").child("latitude").val();
var _lng =
data.child("address").child("longitude").val();
var _name = data.child("name").val();

if (_companyID == userUID) {
arrLocWarehouse[_id] = [_name, _lat, _lng];
initMarker();
}
});

}
```

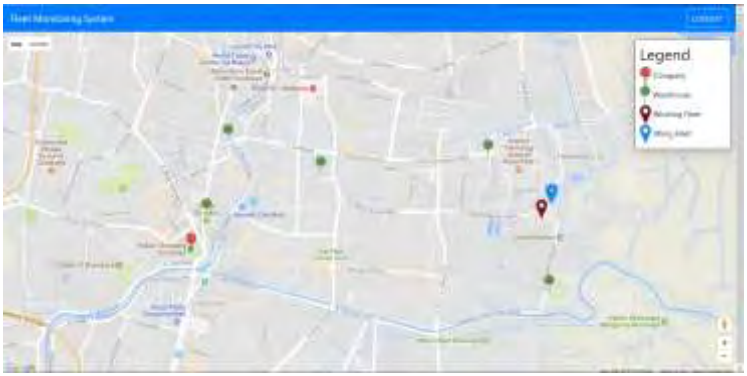
Gambar 5.163 US-AS5-1: Mengambil kordinat lokasi gudang

```
function initMarker() {

  var map = new
google.maps.Map(document.getElementById('map'), {
    zoom: 12,
    center: new google.maps.LatLng(companyLatLng),
    mapTypeId: google.maps.MapTypeId.ROADMAP
  });
  var infowindow = new google.maps.InfoWindow();
  var marker, i, j;

  //Set Company marker
  marker = new google.maps.Marker({
    position: new
google.maps.LatLng(companyLatLng),
    type: 'company',
    map: map,
    icon: {
      url: "assets/images/map_company.png",
      scaledSize: new google.maps.Size(21, 40)
    }
  });

  google.maps.event.addListener(marker, 'click',
(function(marker, i) {
    return function() {
      infowindow.setContent(companyName);
      infowindow.open(map, marker);
    }
  })(marker, i));


  // Set Fleet Marker idling
  for (i = 0; i < arrEmailFleet1.length; i++) {
    marker = new google.maps.Marker({
      position: new
google.maps.LatLng(arrLocFleet1[arrEmailFleet1[i]][
1], arrLocFleet1[arrEmailFleet1[i]][2]),
      type: 'fleetIdling',
      map: map,
      icon: icons['fleetIdling'].icon
    });
    google.maps.event.addListener(marker, 'click',
(function(marker, i) {
      return function() {

infowindow.setContent(arrLocFleet1[arrEmailFleet1[i
]][0]);
        infowindow.open(map, marker);
```

```
      }
    })(marker, i));
  }

  //Set Fleet Marker working
  for (i = 0; i < arrEmailFleet2.length; i++) {
    marker = new google.maps.Marker({
      position: new
google.maps.LatLng(arrLocFleet2[arrEmailFleet2[i]][
1], arrLocFleet2[arrEmailFleet2[i]][2]),
      type: 'fleetWorking',
      map: map,
      icon: icons['fleetWorking'].icon
    });

    google.maps.event.addListener(marker, 'click',
(function(marker, i) {
      return function() {

infowindow.setContent(arrLocFleet2[arrEmailFleet2[i
]][0]);
        infowindow.open(map, marker);
      }
    })(marker, i));
  }

  //Set Warehouse Marker
  for (j = 0; j < arrIDWarehouse.length; j++) {
    marker = new google.maps.Marker({
      position: new
google.maps.LatLng(arrLocWarehouse[arrIDWarehouse[j
]][1], arrLocWarehouse[arrIDWarehouse[j]][2]),
      type: 'warehouse',
      map: map,
      icon: icons['warehouse'].icon
    });

    google.maps.event.addListener(marker, 'click',
(function(marker, j) {
      return function() {

infowindow.setContent(arrLocWarehouse[arrIDWarehous
e[j]][0]);
        infowindow.open(map, marker);
      }
    })(marker, j));
  }

}
```

Gambar 5.164 US-WM2-1: Membuat marker-marker lokasi

Tampilan persebaran lokasi adalah seperti gambar 5.165.



Gambar 5.165 US-AS5-1: Edit nomor plat truk sopir

5.4.1.3 US-WM2-2 Sopir Sedang Bekerja /Beristirahat

Data lokasi armada dipisahkan dalam 2 list, yakni arrLocFleet1 untuk menyimpan daftar armada yang sedang tidak bekerja dan arrLocFleet2 untuk menyimpan daftar armada yang bekerja. Saat peta dinisiasi, arrayLocFleet1 menggunakan marker berwarna biru dan arrayLocFleet2 menggunakan marker berwarna merah ( ) seperti pada gambar 5.165.

BAB 6
UJI COBA

Pada baba ini dijelaskan hasil uji coba aplikasi yang sudah berhasil di implementasikan baik uji coba fungsional sesuai daftar user story pada table 4-3, 4-4, dan 4-5, maupun uji non-fungsional yakni performa aplikasi.

6.1 Uji Coba Fungsional

Hasil dari uji coba fungsional merupakan langkah yang dilakukan untuk menguji aplikasi dapat menjalankan semua fungsi yang didefinisikan diawal sesuai dengan Tabel 4-2, 4-3, dan 4-4 . Hasil *test case* dapat dilihat pada Tabel 6.1. Status yang menggunakan teks berwarna hijau berarti sesuai dengan Tabel User Story, sedangkan berwarna merah berarti tidak sesuai.

Tabel 6.1 Uji coba fungsional

User Story	Deskripsi	Skenario	Status
Smartphone Manajemen Perusahaan			
US-AM1-1	*Sign Up*	*Sign up* dengan akun *email* baru	Berhasil
		Sign up dengan *email* yang sudah terdaftar	Gagal
US-AM1-2	*Sign In*	*Sign in* dengan *email* dummy	Gagal
		Sign in email benar *password* salah	Gagal

		Sign in dengan akun armada	Berhasil
US-AM1-3	Mendaftarkan aku armada	Mendaftarkan dengan beberapa informasi tidak disertakan	Gagal
		Mendaftarkan dengan semua informasi disertakan	Berhasil
		Mendaftarkan dengan *email* yang sudah ada	Gagal
US-AM1-4	Mengedit informasi akun perusahaan	Mengedit *email* dan *password*	Berhasil
		Mengedit informasi pendukung	Berhasil
US-AM1-5	Mengedit informasi akun armada	Mengedit informasi pendukung	Berhasil
US-AM2-1	Membuat penugasan baru	Membuat penugasan dengan semua data diisikan	Berhasil
		Membuat penugasan dengan	Gagal

		beberapa data tidak diisikan	
US-AM2-2	Melihat lokasi dan tugas sopir	Melihat sopir di peta	Berhasil
		Melihat sopir dalam *list*	Berhasil
US-AM2-3	Membatalkan pengiriman barang	Membatalkan 1 penugasan	Berhasil
US-AM2-4	Melihat penugasan yang sedang berjalan	Melihat daftar tugas yang belum diselesaikan	Berhasil
		Melihat detail deskripsi tugas yang belum diselesaikan	Berhasil
		Melihat detail riwayat lokasi tugas yang belum diselesaikan	Berhasil
US-AM2-5	Melihat penugasan yang sudah berakhir	Melihat daftar tugas yang sudah diselesaikan	Berhasil
		Melihat detail deskripsi tugas yang sudah diselesaikan	Berhasil
		Melihat detail rute tugas yang sudah diselesaikan	Berhasil

US-AM3-1	Menerima laporan barang yang sampai	Laporan yang sampai adalah hasil konfirmasi pelanggan	Berhasil
US-AM4-1	Membuat penugasan lanjutan pada sopir yang sama	Penugasan sesuai dengan jumlah barang yang belum sampai	Berhasil
US-AM4-2	Membuat penugasan lanjutan pada sopir yang berbeda	Penugasan sesuai dengan barang yang belum sampai	Berhasil
US-AM4-3	Mengalihkan penugasan ke sopir lain	Mengalihkan penugasan dari sopir yang mengajukan ke sopir lain	Berhasil
US-AM5-1	Membuat, memodifikasi, dan penghapus gudang	Membuat gudang baru	Berhasil
		Mengubah nama gudang	Berhasil
		Mengubah lokasi gudang	Berhasil
		Menghapus gudang	Berhasil
US-AM6-1	*Reset Password*	Meminta *reset password*	Berhasil
US-AM7-1	Melihat kinerja tiap armada	Melihat jumlah tugas	Berhasil

		yang berhasil dan gagal dikerjakan armada	
Smartphone Sopir Armada			
US-AS1-1	*Sign In*	*Sign in* dengan *email* dan *password* yang benar	Berhasil
		Sign in dengan akun *dummy*	Gagal
		Sign in yang pertama kali (*Sign up*)	Berhasil
		Sign in dengan akun perusahaan	Gagal
US-AS1-2	Menerima dan melihat tugas yang diberikan	Melihat daftar tugas yang belum selesai	Berhasil
		Melihat detail tugas yang belum selesai	Berhasil
US-AS1-3	Mengajukan pengalihan ke sopir lain	Mengajukan 1 tugas ke sopir lain	Berhasil
US-AS2-1	Melihat tugas yang sudah diselesaikan	Melihat daftar tugas yang sudah diselesaikan	Berhasil
		Melihat detail deskripsi tugas yang	Berhasil

		sudah diselesaikan	
		Melihat detail rute tugas yang sudah diselesaikan	Berhasil
US-AS3-1	Menyimpan riwayat lokasi saat melaksanakan pengiriman barang	Menyimpan riwayat lokasi di *database* dan meletakkan *marker*-nya di peta	Berhasil
US-AS3-2	Mengisi form penyelesaian tugas	Mengisi dan mengirimkan penyelesaian tugas dengan token yang salah	Berhasil
		Mengisi dan mengirimkan tugas dengan token yang benar	Berhasil
US-AS3-3	Mengajukan pembatalan pengantaran tugas yang sedang dikerjakan	Mengajukan pembatalan semua tugas yang sedang dikerjakan dalam 1 *departure*	Berhasil
		Membatalkan saat ada tugas yang sudah selesai	Berhasil
US-AS3-4	Menerima konfirmasi	Melihat apakah tugas	Berhasil

	tugas belum berhasil diselesaikan	yang belum selesai tetap dikerjakan atau sudah dialihkan	
US-AS3-5	Menerima pemberitahuan pembatalan tugas	Menerima pesan untuk tidak mengerjakan tugas	Berhasil
US-AS4-1	Mengirim lokasi ke *server*	Mengirimkan kordinat lokasi terkini ke *server*	Berhasil
US-AS4-2	Melihat lokasi terkini, lokasi gudang, dan lokasi perusahaan	Melihat peta yang menunjukkan lokasi sat ini, lokasi gudang, dan lokasi perusahaan	Berhasil
US-AS5-1	Mengedit informasi akun	*Mengedit email* dan *password* akun	Berhasil
		Mengedit informasi pendukung akun	Berhasil
US-AS6-1	*Reset Password*	Mengajukan *reset password*	Berhasil
Website Manajemen Perusahaan			
US-WM1-1	*Login Website*	Login dengan akun dummy	Gagal

		Login dengan email dan password perusahaan	Berhasil
		Login dengan password yang salah	Gagal
		Login dengan email armada	Gagal
US-WM2-1	Melihat lokasi seluruh armada	Ada marker alamat perusahaan, gudang dan lokasi sopir	Berhasil
		Marker ditampilkan secara realtime	Berhasil
US-WM2-2	Sopir sedang bekerja/istirahat	Ada perbedaan marker sopir yang bekerja dan tidak bekerja	Berhasil
		Perubahan warna marker saat sopir sudah tidak bekerja atau memulai bekerja	Berhasil

6.2 Uji Coba Non-Fungsional

6.2.1 Kecocokan di Berbagai Sistem Operasi

Pengujian dilakukan pada emulator Android Virtual Devices yang disediakan oleh Android Studio. Pengujian ini dilakukan pada smartphone yang sama yaitu Google Pixel dengan versi android yang berbeda-beda seperti pada tabel 6-1. Aplikasi yang diuji adalah:

1. Fleemo Company
2. Fleemo Driver

Tabel 6.2 Pengujian aplikasi di berbagai sistem operasi

Versi Android	Aplikasi	
	1	2
JellyBean API 16	X	X
JellyBean API 18	X	X
KitKat API 19	✓	✓
Lollipop API 22	✓	✓
Marshmallow API 23	✓	✓
Nougat API 24	✓	✓
Nougat API 25	✓	✓
Oreo API 26	✓	✓
API 27	✓	✓

Pada sistem operasi dengan API 16 dan 18, aplikasi bisa diinstal, dijalankan, dan bisa mengambil data dari Firebase. Namun aplikasi tidak mengenali fungsi Firebase yakni FirebaseAuth, sehingga saat login atau sign up aplikasi menjadi error dan *force close* dengan kode:

```
java.lang.NullPointerException at
com.google.firebase.auth.FirebaseAuth.signInWithE
mailAndPassword(Unknown Source)
```

Gambar 6.1 Pesan Error ujicoba Android API <18

6.2.2 Pengujian Saat Tidak Ada Internet

Ada 2 pengujian yang dilakukan. Apabila tidak ada internet saat sudah berada di *activity* utama HomeAcitivity, aplikasi masih bisa dijalankan dan semua data yang ada di setiap fragment yang terkait dengan HomeAcitivity masih ditampilkan, hanya saja tidak bisa memperbaharui dengan data terbaru. Saat aktivitias baru dibuka dan data belum dimuat sebelumnya, aplikasi akan *force close*.

Apabila tidak ada internet saat aplikasi belum dibuka sama sekali, maka saat dibuka aplikasi hanya sampai pada *activity* pembuka FlashActivity dan tidak masuk ke *activity* lainnya. Hal ini disebabkan karena pada FlashActivity mengecek user yang login dan mengecek akses lokasi. Karena tidak berhasil, FlashActivity tidak selesai.

6.2.3 Pengujian Saat Lokasi Tidak Diaktifkan

Pada FleemoCompany, saat lokasi tidak diaktifkan aplikasi tetap dapat dijalankan karena aplikasi tidak membutuhkan lokasi terkininya dikirim ke server. Namun saat *activity* yang membutuhkan lokasi terkini seperti Set_Account_Map dijalankan, Map akan langsung menampilkan lokasi pada Kordinat (0,0). Saat kordinat itu dipilih, aplikasi akan *force close* karena Geocoder yang mengkonversi kordinat menjadi alamat tidak dapat mengkonversikan alamat tersebut. Tapi tidak ada masalah jika user dengan manual meletakkan posisi *pin set location*.

Pada Fleemo Driver, saat lokasi tidak diaktifkan aplikasi tetap dapat dijalankan. Namun pada HomeActivity fragment MapActivity, peta menampilkan posisi peta pada kordinat (0,0) dan tidak memberikan perubahan saat tombol reCenter ditekan. Aplikasi juga tidak bisa mengirimkan lokasi terkininya. Saat

lokasi kembali diaktifkan dan tombol reCenter ditekan, maka aplikasi langsung menampilkan marker armada pada posisinya saat ini.

6.2.4 Kecepatan Aplikasi Android

Kecepatan aplikasi yang diuji adalah kecepatan saat mengirim dan menerima data dengan cara menghitung lama perpindahan dari *activity* 1 ke *activity* lain maupun dengan cara menghitung durasi 1 *activity* mengirimkan data ke database. Pada Tabel 6-2 dan 6-3. Pengujian dilakukan di beberapa perangkat yang berbeda menggunakan Emulator yang disediakan oleh Android Studio. Spesifikasi emulator yang dipakai adalah

1. Nexus 5 (Processor:4 & RAM: 1536MB)
2. Nexus S (Processor:4 & RAM: 343MB)
3. Pixel (Processor:2 & RAM:1536MB)
4. Pixel 2XL (Processor:2 & RAM: 343MB)

Beberapa *activity* utama yang butuh diuji adalah:

6.2.4.1 Fleemo Company

Tabel 6.3 Uji kecepatan fleemo company

Aktivitas	**Durasi** (detik)			
	1	**2**	**3**	**4**
FlashActivity→SignInActivity	2, 285	3,079	2,136	2,299
FlashActivity→SignUpActivity	0,384	0,204	0,789	1,290
SignInActivity→HomeActivity	0,540	1,585	3,142	3,582
SignUpActivity→HomeActivity	0,968	0,441	1,083	1,843
HomeActivity→OnGoing Detail	0.753	0,656	1,557	2,165
HomeActivity→Finished Detail	0,366	0,405	1,152	2,119

HomeActivity→Company_Edit_Name	0,239	0,172	0,952	1,585
HomeActivity→FleetAccountDetail	0,672	0,468	1,315	2,142
HomeActivity→FleetRegistration	0,329	0,208	0,962	1,964
HomeActivity→MessageActivity	0,286	0,156	1,106	2,22
MessageActivity→MessageDetail	0,463	0,369	1,467	1,556
FinishedDetail→ ReworkTask	0,370	0,254	0,896	2,389
ReworkTask→HomeActivity	0,529	0,332	1,023	2,013

6.2.4.2 Fleemo Driver

Tabel 6.4 Uji kecepatan fleemo driver

Aktivitas	**Durasi** (detik)			
	1	**2**	**3**	**4**
FlashActivity	1,749	5,902	2,66	2,746
FlashActivity→SignInActivity	0,734	3,281	1,772	3,164
SignInActivity→HomeActivity (Sign In)	2,595	3,882	4,126	2,484
SignInActivity→HomeActivity (Sign Up)	0,846	0,861	4,185	2,636
HomeActivity→NotFinished_Detail	0,740	0,680	1,395	2,610
HomeActivity→Finished_Detail	0,390	0,248	1,118	1,744
HomeActivity→EditFullName	0,278	0,097	0,985	1,876
HomeActivity→SignInActivity	0,311	0,148	0,899	1,556
HomeActivity→TaskOngoing	1,438	6,926	1,176	1,948

TaskOngoing→CompletionActivity	0,563	0,350	1,370	1,841
CompletionActivity→TaskOngoing	1,289	1,395	1,219	4,538
CompletionActivity→HomeActivity	13,545	6,630	28,781	46,64

6.2.5 Kecocokan Website di Berbagai Browser

Pada uji ini, Fleemo Website akan dijalankan dalam berbagai *web browser* bedasarkan Tabel 6-3. Dari web browser yang digunakan, Internet Explorer menganggap halaman tersebut harus diblok, sehingga blok konten harus dibatalkan agar halaman terbuka. Saat dashboard terbuka, web *not responding*. Pada Ms. Edge, pada halaman login, web selalu memberikan peringatan email/password salah meskipun email terdaftar di Firebase. Akibatnya, user tidak bisa masuk ke dashboard.html

Tabel 6.5 Kecocokan website di berbagai browser

Browser	**Halaman Web**	
	login.html	**Dashboard.html**
Mozilla Firefox	mendukung	mendukung
Chrome	mendukung	mendukung
Internet Explorer	Tidak mendukung (saat login berhasil, langsung kembali ke halaman login.html)	-
Ms. Edge	Tidak mendukung (tidak membaca firebase)	-
Puffin	mendukung	mendukung

(Halaman Sengaja Dikosongkan)

BAB 7
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan kesimpulan dan saran dari seluruh pengerjaan Tugas Akhir. Kesimpulan dan saran diharapkan dapat berguna untuk proses pengembangan aplikasi selanjutnya.

7.1 Kesimpulan

Berdasarkan pengerjan Tugas Akhir ini, didapatkan beberapa kesimpulan sebagai berikut:

1. Sistem Pemantau Armada terdiri dari 3 aplikasi yakni aplikasi berbasis Android untuk digunakan oleh manajemen perusahaan yang bernama *Fleemo Company*, aplikasi berbasis Android untuk digunakan oleh sopir armada yang bernama *Fleemo Driver*, dan website yang dipakai perusahaan yang bernama Fleemo Website untuk bisa melihat lokasi persebaran armada dan gudang-gudang miliknya dalam tampilan layar besar.
2. Pembuatan aplikasi berhasil dibuat menggunakan Database cloud yang disediakan oleh Google Firebase dengan JSON sebagai format pertukaran data. Pembangunan aplikasi menggunakan Firebase cukup sederhana karena semua fitur yang dibutuhkan dalam pembuatan aplikasi sudah disediakan oleh Firebase. Keamanan datanya juga cukup baik karena menggunakan API Google sehingga yang bisa mengakses adalah aplikasi yang meiliki API tersebut. Namun Firebase cukup memakan memori dan menyimpan banyak cache. Hal ini membuat perangkat terkadang lambat dan force close.
3. Dari sisi aplikasi perusahaan, user mengisikan data alamat perusahaan (kordinat lokasi), dan mendaftarkan gudang-

gudang miliknya yang berisi nama dan alamat (kordinat lokasi). Kordinat diambil menggunakan Google Map yakni mengambil titik tengah peta pada layar smartphone. Data ini akan digunakan aplikasi Pemantau Armada untuk meletakkan *marker* lokasi perusahaan dan gudang di peta.

4. Dari sisi aplikasi armada, aplikasi secara otomatis menentukan kordinat lokasi kendaraan saat ini menggunakan fungsi android yakni *Location Listener* lalu menyimpannya di database. Kordinat yang diambil menggunakan aplikasi *Fleemo Driver* ini akan secara otomatis merubah marker pada peta di ketiga aplikasi saat armada berpindah lokasi.
5. Hasil kinerja armada hanya dapat dinilai dari berhasil atau tidaknya sopir mengantarkan 1 penugasan yang diberikan. Manajemen perusahaan bisa melihat performa mereka dari berapa tugas yang berhasil diselesaikan dan berapa yang gagal. Konsep untuk menilai kinerja beberapa sopir pada 1 penugasan yang sama cukup sulit dilakukan karena barang yang diantarkan dalam setiap penugasan kemungkinan besar selalu berbeda, lokasi yang ditempuh sopir berbeda, dan lokasi gudang berbeda.
6. Aplikasi Android manajemen perusahaan dapat digunakan untuk memberikan penugasan-penugasan baru pada sopir. Saat tugas berhasil dibuat, maka secara *realtime* sopir dapat menerima tugas tersebut untuk dikerjakan. Manajemen perusahaan juga bisa melihat apakah tugas yang diberikan belum dikerjakan, sedang dikerjakan, atau sudah selesai.
7. Saat sopir selesai mengantarkan barangnya, ia perlu mengisi *form* penyelesaian tugas agar perusahaan tahu tiap penugasan yang sudah selesai. Penyelesaian ini dikonfirmasi langsung dari pelanggan dimana pelanggan

akan mengisikan TOKEN yang hanya diketahui oleh manajemen perusahaan dan pelanggan di *form* tersebut.

7.2 Saran

Pada bagian ini akan dipaparkan kekurangan dari aplikasi *Fleet Monitoring System* untuk penelitian selanjutnya agar diperbaiki dan dikembangkan lagi. Berikut adalah masukan untuk penelitian selanjutnya:

1. Saat ini aplikasi tidak memiliki fitur Vehicle Routing Problem sehingga sopir tidak bisa pergi dengan rekomendasi jalur yang terbaik. Bila butuh, sopir harus menggunakan aplikasi lain seperti GoogleMap. Dengan adanya fitur ini, sopir dapat langsung mendapatkan rekomendasi rute tanpa perlu berpindah aplikasi. Berpindah aplikasi rentan menutup aplikasi tersebut sehingga riwayat lokasi tidak tercatat.
2. Perlunya notifikasi yang dapat diterima baik perusahaan atau sopir. Bagi sopir, notifikasi perlu seperti pada saat menerima penugasan baru, penugasan dibatalkan, respon atas pembatalan tugas yang diajukan, dan notifikasi apakah tugas yang tidak berhasil diselesaikan akan tetap dikerjakan oleh sopir tersebut atau dialihkan ke sopir lain. Bagi perusahaan, notifikasi perlu saat penugasan sopir diselesaikan sehingga perusahaan dengan cepat bisa tahu apakah berhasil atau gagal. Saat gagal perusahaan bisa cepat memberikan *Rework Task*. Notifikasi juga perlu agar perusahaan dengan cepat tahu apakah ada pengajuan pengalihan tugas.
3. *Rule* pengambilan data pada Firebase perlu diperketat untuk menjaga agar orang-orang tidak bebas mengakses semua ranting database. Saat ini rule yang dipakai adalah Semua bisa membaca dan menulis.

4. Web memiliki fitur yang sama dengan aplikasi Fleemo Company, karena mengakses data pada *computer* lebih mudah dibandingkan pada *smartphone*.
5. Melakukan iterasi berikutnya dengan metode *Google design sprint* agar aplikasi semakin sempurna (*continuous improvement*) lalu melakukan pengujian pada calon pengguna aplikasi agar kebutuhan mereka benar-benar tercapai.

DAFTAR PUSTAKAs

[1] English Oxford Living Dictionaries, "Oxford dictionaries," [Online]. Available: https://en.oxforddictionaries.com/definition/smartphone. [Accessed 25 January 2017].

[2] "Number of smarphone users worldwide from 2014 to 2020 (in billions)," Statista, [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed 25 January 2017].

[3] "Global market share held by the leading smartphone operating system in sales to end users from 1st quarter 2009 to 3rd quarter 2016," statista, [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/. [Accessed 25 January 2017].

[4] pricebook, "Daftar Harga iOS Januari 2017," [Online]. Available: http://www.pricebook.co.id/smartphone?operating_system=iOS. [Accessed 25 January 2017].

[5] Apple, "Choosing a Membership," [Online]. Available: https://developer.apple.com/support/compare-memberships/. [Accessed 2015 January 2015].

[6] Trackimo, "Best 3G GPS Tracker Devices - Trackimo," [Online]. Available: http://trackimo.com/. [Accessed 15 February 2017].

[7] Teletrac Navman, "GPS Fleet Management - Teletrac Navman," [Online]. Available: http://www.teletracnavman.com/. [Accessed 15 February 2017].

[8] Rhino Fleet Tracking, "Fleet Tracking System | Fleet Tracker Devices | Fleet Trackers | Rhino," [Online]. Available: http://www.rhinofleettracking.com/. [Accessed 15 February 2017].

[9] Android , "Platform architecture," [Online]. Available: https://developer.android.com/guide/platform/index.html. [Accessed 31 January 2017].

[10] FIrebase, "Firebase Feature," [Online]. Available: https://firebase.google.com/features/. [Accessed 30 January 2017].

[11] Firebase, "Structure Your Database | Firebase," [Online]. Available: https://firebase.google.com/docs/database/android/structure-data. [Accessed 30 January 2017].

[12] Firebase, "Read and Write Data," [Online]. Available: https://firebase.google.com/docs/database/android/read-and-write. [Accessed 30 January 2017].

[13] Firebase, "Work with List of Data," [Online]. Available: https://firebase.google.com/docs/database/android/lists-of-data. [Accessed 30 January 2017].

[14] Google, "Google Maps Android API Utility Setup," [Online]. Available:

https://developers.google.com/maps/documentation/android-api/utility/setup. [Accessed 30 January 2017].

[15] base36, "Agile & Watefall Methodologies - A Side-By-Side Comparison," [Online]. Available: http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison/. [Accessed 27 February 2017].

[16] SmartSheet, "Full Comparison: Agile vs Scrum vs Waterfall vs Kanban," [Online]. Available: https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban. [Accessed 2017 February 2017].

[17] Google, "Design Sprint | Google Developers," [Online]. Available: https://developers.google.com/design-sprint/product/. [Accessed 8 February 2017].

[18] J. Rubin and D. Chisnell, Handbook of Usability Testing | Second Edition, IndianaPolis: Wiley Publishing, Inc., 2008.

[19] Canvas Flip, "User Testing v/s Usability testing," [Online]. Available: http://blog.canvasflip.com/index.php/2016/06/12/user-testing-vs-usability-testing/. [Accessed 7 February 2017].

[20] J. Preece, A Guide to Usability: Human Factors in Compting, Addison Wesley, the Open University, 1993.

[21] A. A. A. Carvalho, "Usability Testing of Educational Software: methods, techniques, and evaluator," no. Usability Testing.

[22] I. Delikostidis, *Methods and Techniques for Field-Based Usability Testing of Mobile Geo-Application,* 2007.

[23] Google, "The "How Might We" Note Taking Method - Understand - Design Sprint Kit," [Online]. Available: https://designsprintkit.withgoogle.com/methods/understand/hmw-directions/. [Accessed 17 Mei 2017].

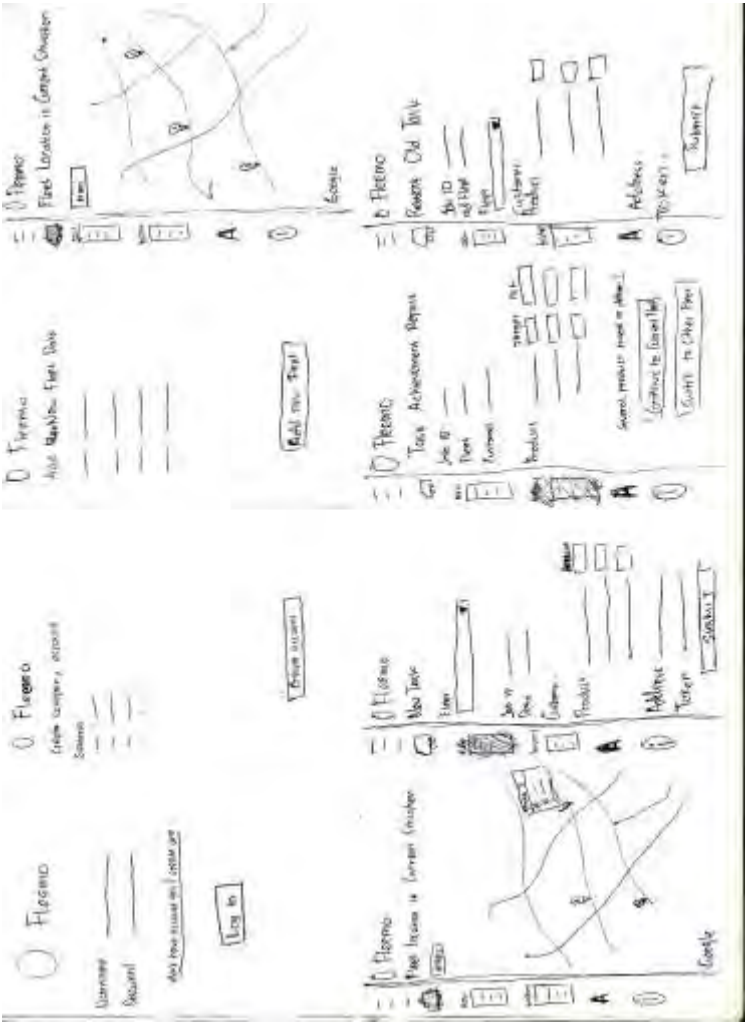
[24] Firebase, "Add Firebase to Your Android Project | Firebase," [Online]. Available: https://firebase.google.com/docs/android/setup. [Accessed 30 January 2017].

[25] D. Peterson, "Kanban Blog," Kanban Blog, 2009. [Online]. Available: http://kanbanblog.com/. [Accessed 27 February 2017].

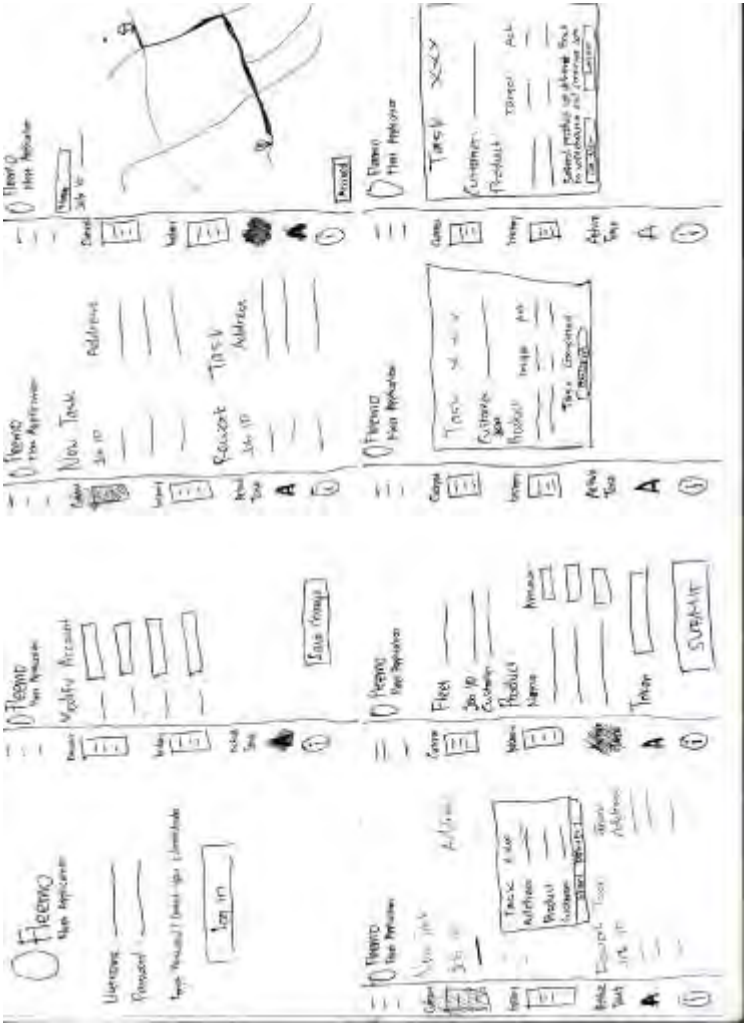
[26] S. W. Ambler, "User Stories: An Agile Introduction," Ambysoft Inc., 2013. [Online]. Available: http://www.agilemodeling.com/artifacts/userStory.htm. [Accessed 21 March 2017].

LAMPIRAN

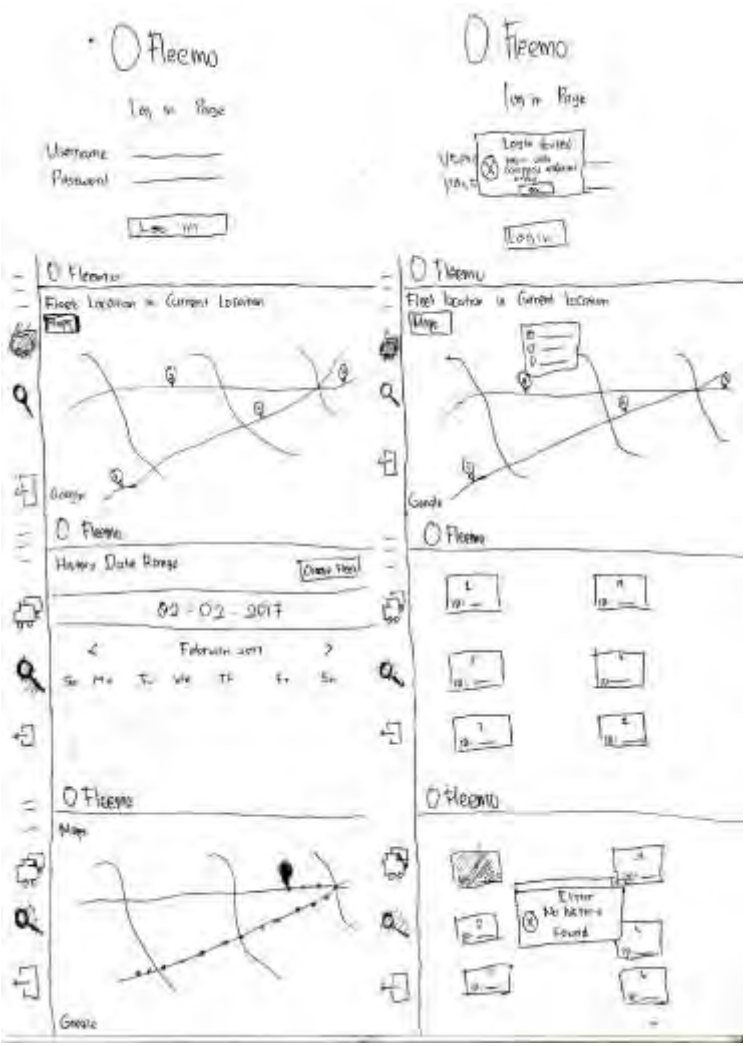
Lampiran 1



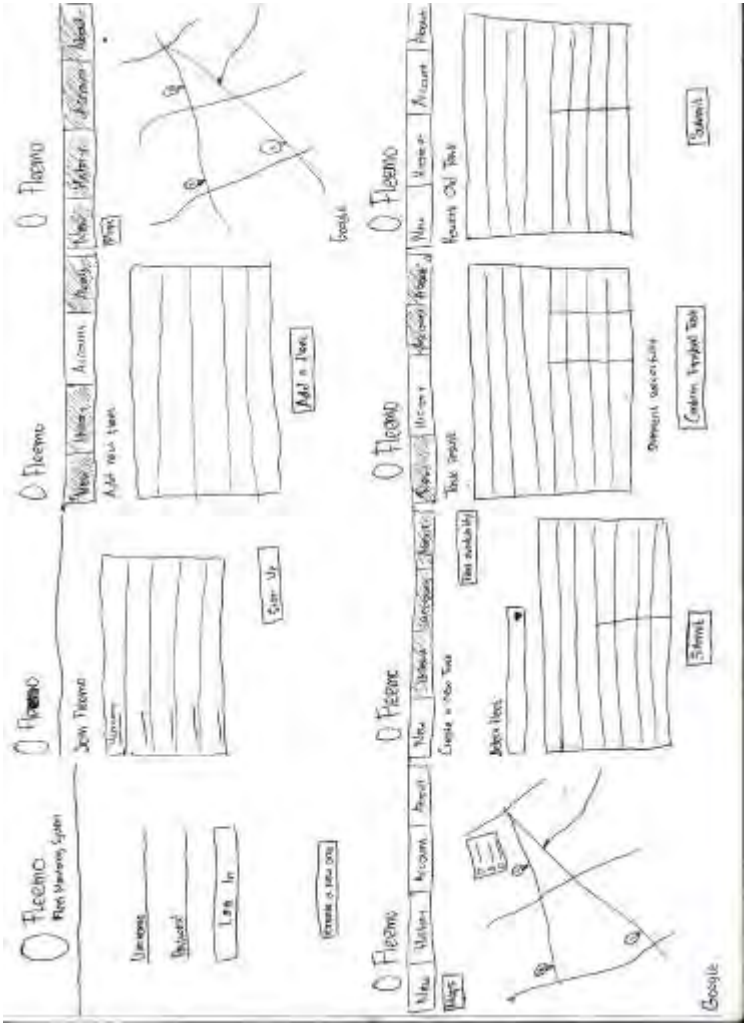
Lampiran 2



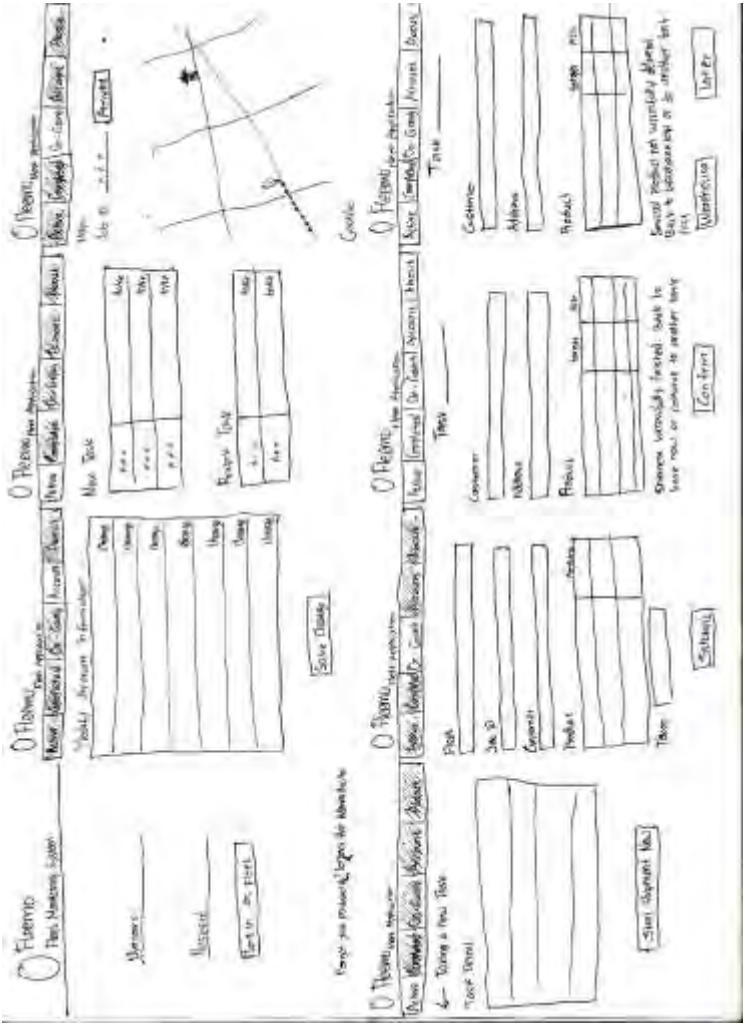
Lampiran 3


Fleemo
Log in Page
Username
Password
Log in
Fleemo
Log in Page
Login
Fleemo
Fleet location in Current location
Map
Fleemo
Fleet location in Current location
Map
Fleemo
History Date Range
Choose Year
02 - 02 - 2017
Fleemo
Fleemo
Map
Google
Fleemo
Error
No history
found

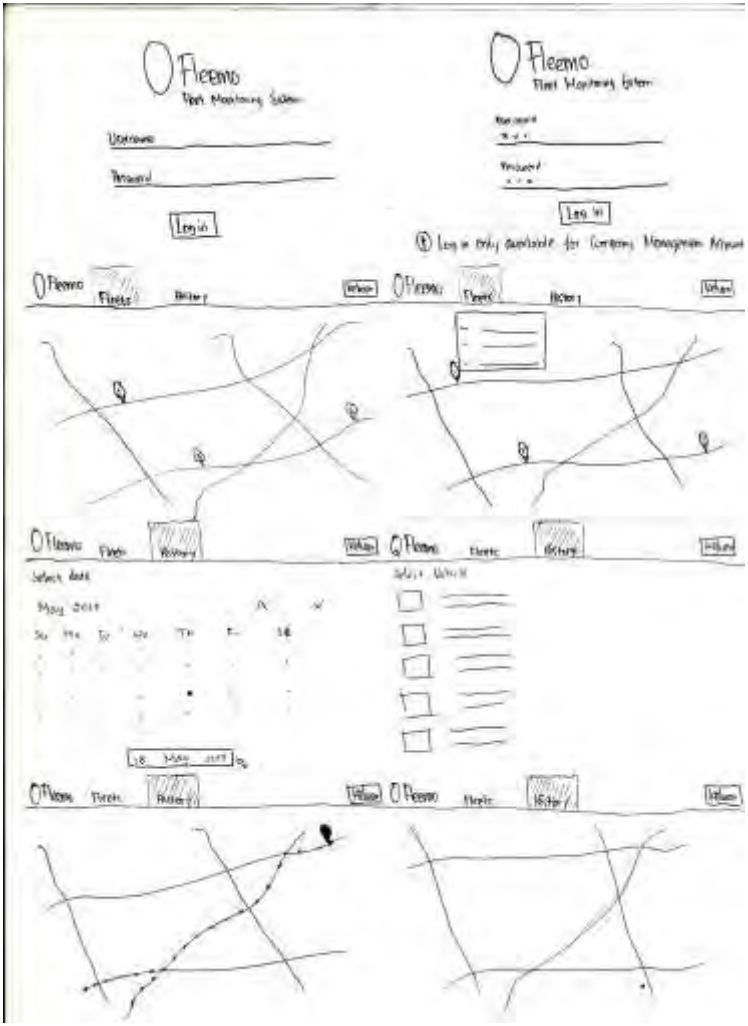
Lampiran 4



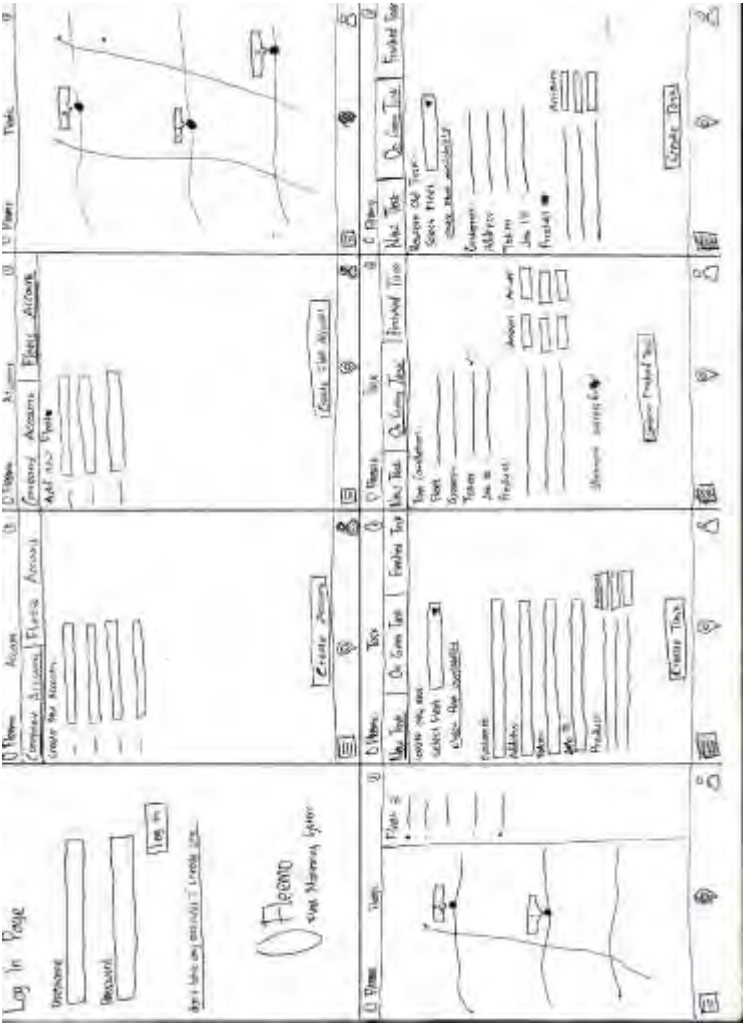
Lampiran 5



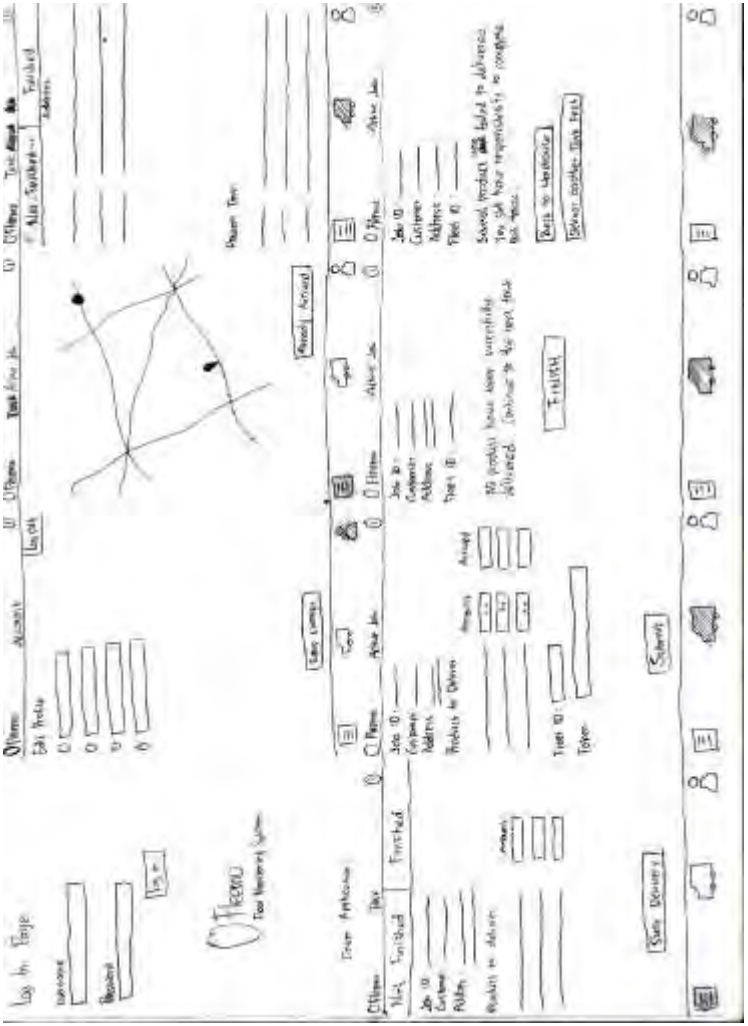
Lampiran 6


Fleemo
Fleet Monitoring System
Username
Password
Login
Fleemo
Fleet Monitoring System
Login
Select date
May 2019

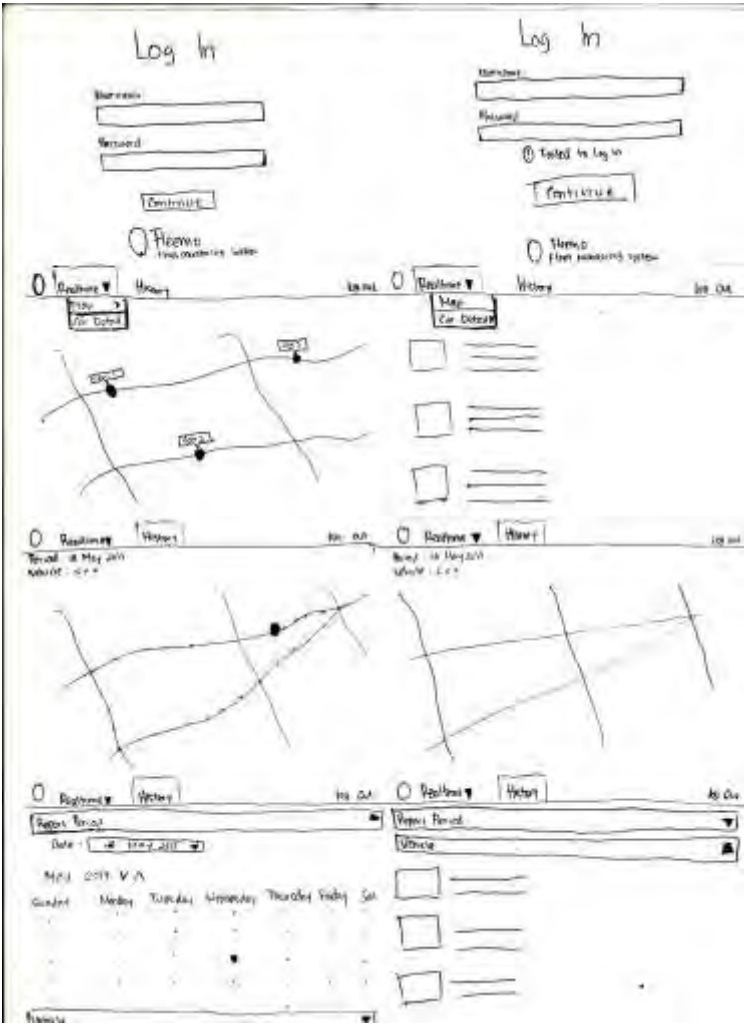
Lampiran 7


Log In Page
Username
Password
Log In
Don't have an account? Create one
Florento
Your Marketing Partner
Create Account
Create Task
Create Project Task
Create Task

Lampiran 8

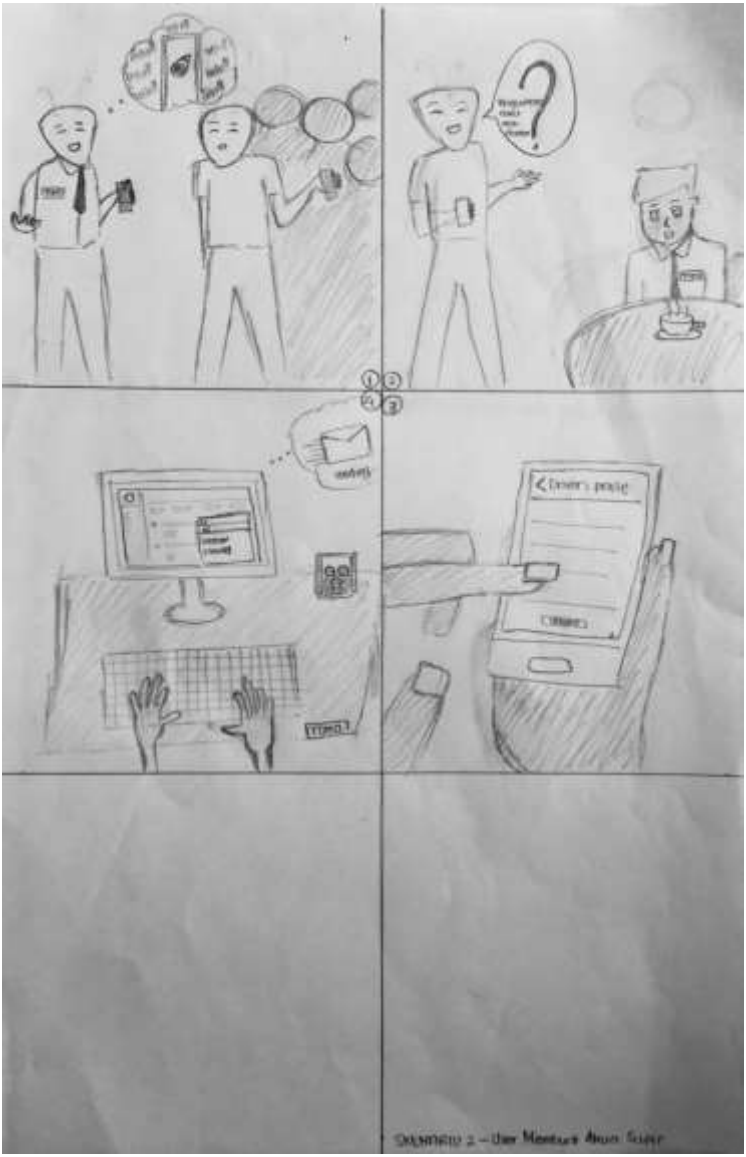


Lampiran 9

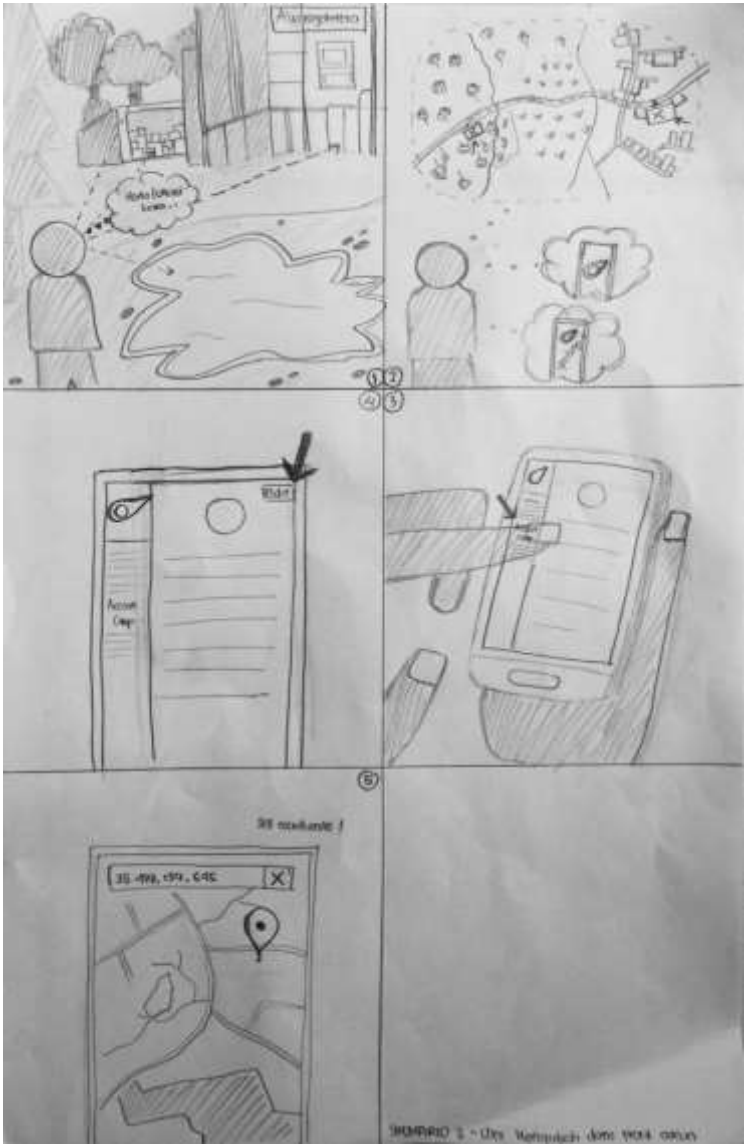

Log In
Log In
Map
Car Detail
Map
Car Detail
Monday
Tuesday
Wednesday
Thursday
Friday
Sat

Lampiran 10

Lampiran 11



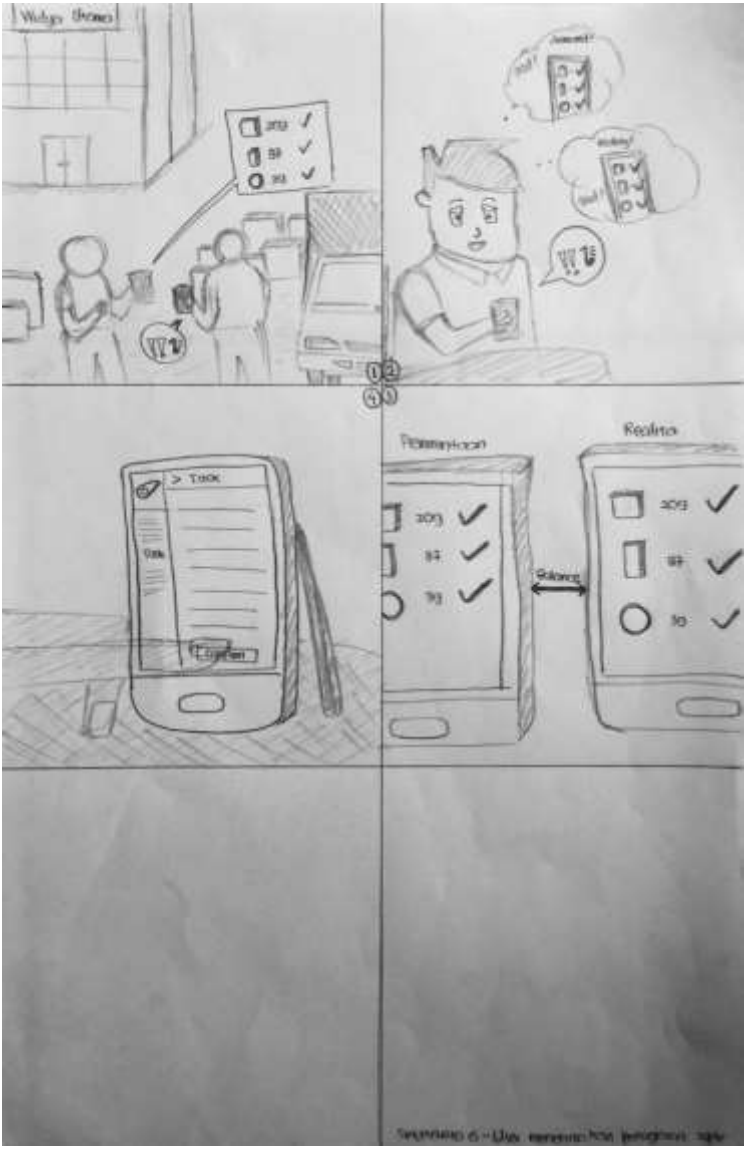
Lampiran 12



Lampiran 13

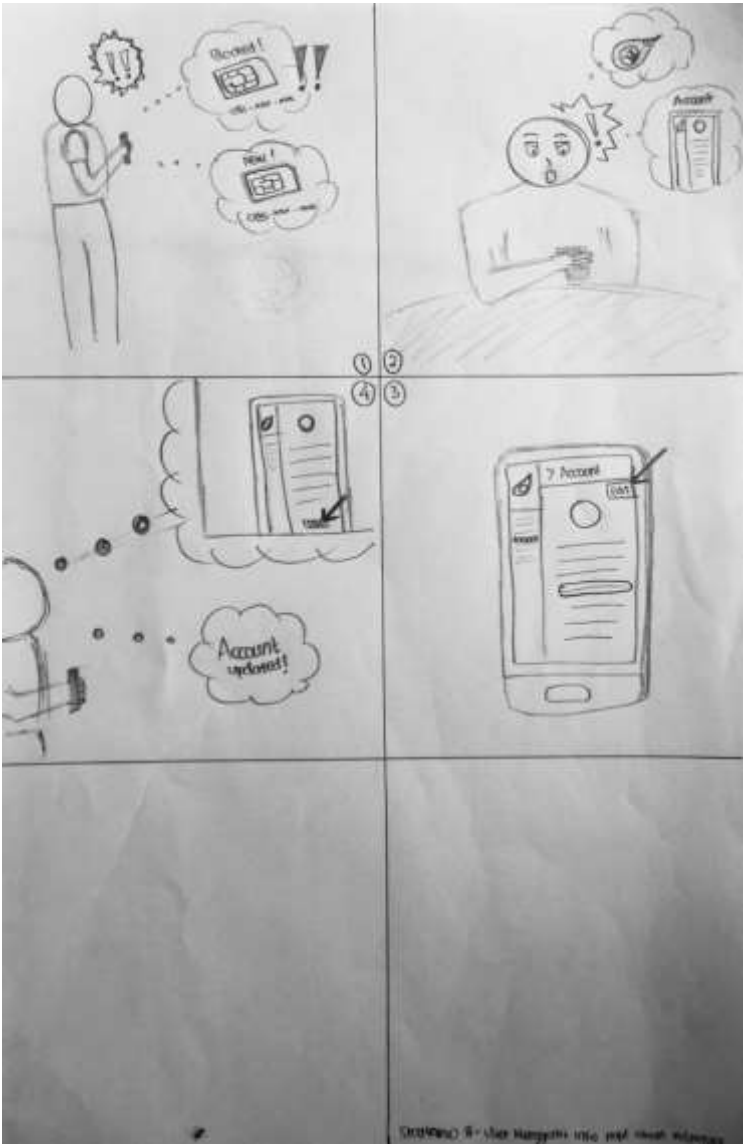
Lampiran 14

Lampiran 15

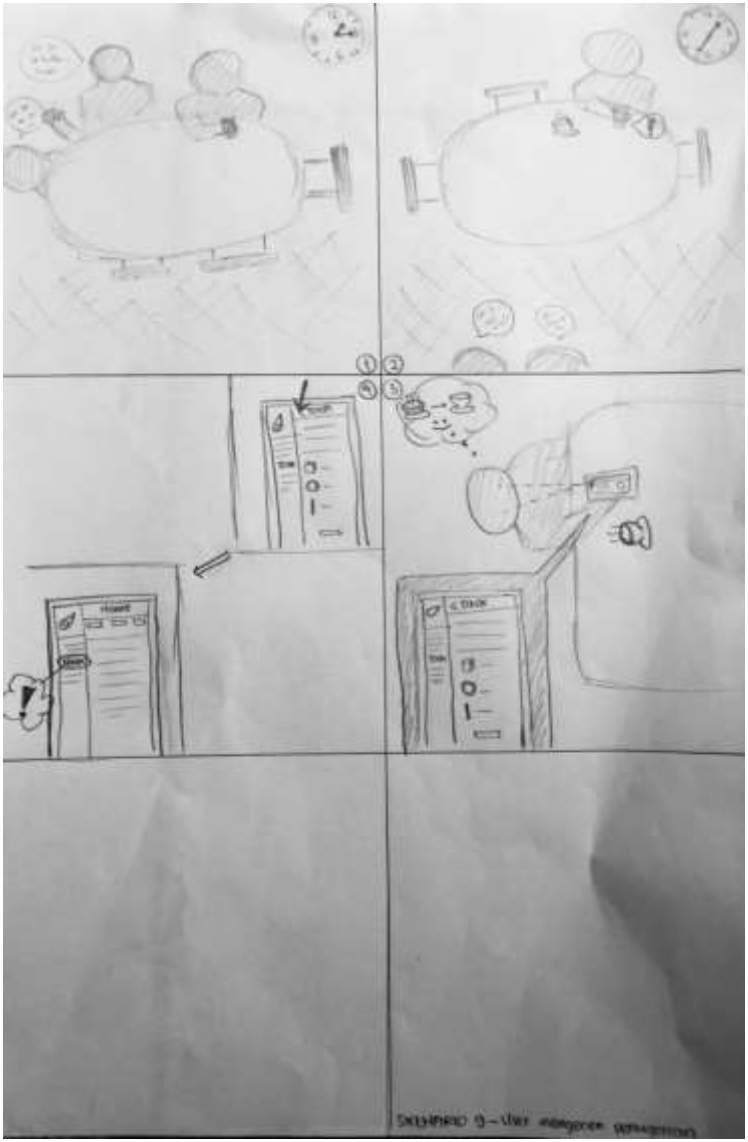


Lampiran 16

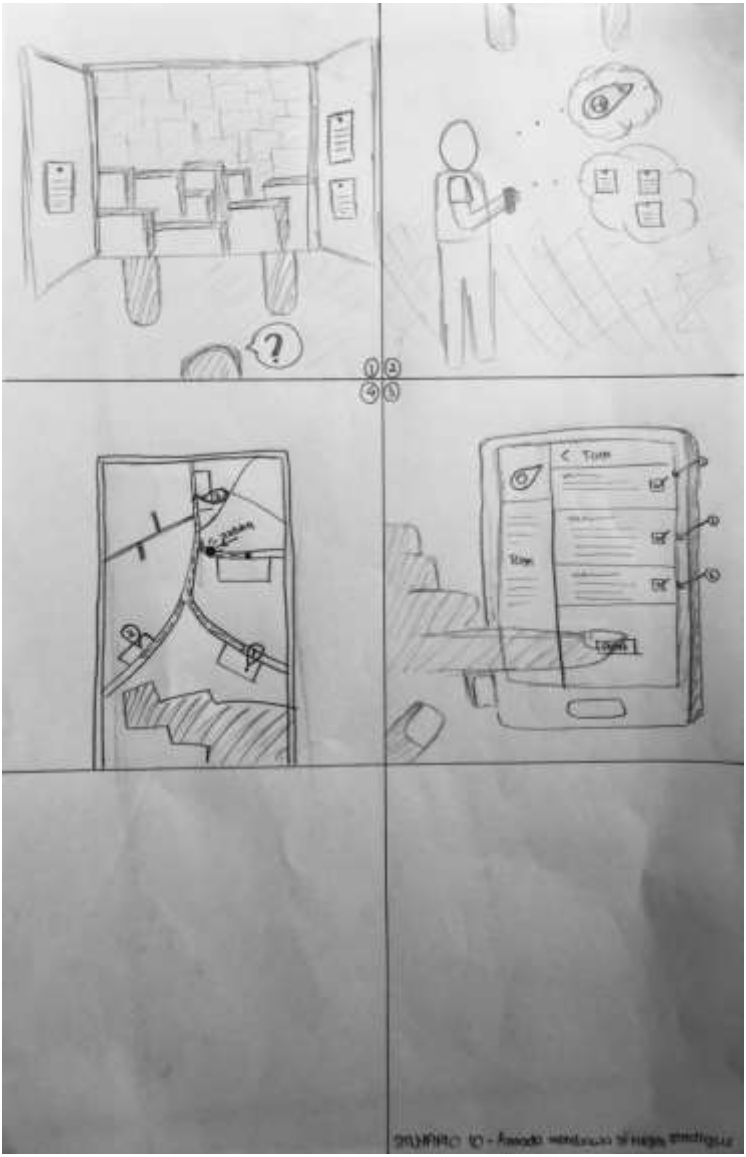
Lampiran 17



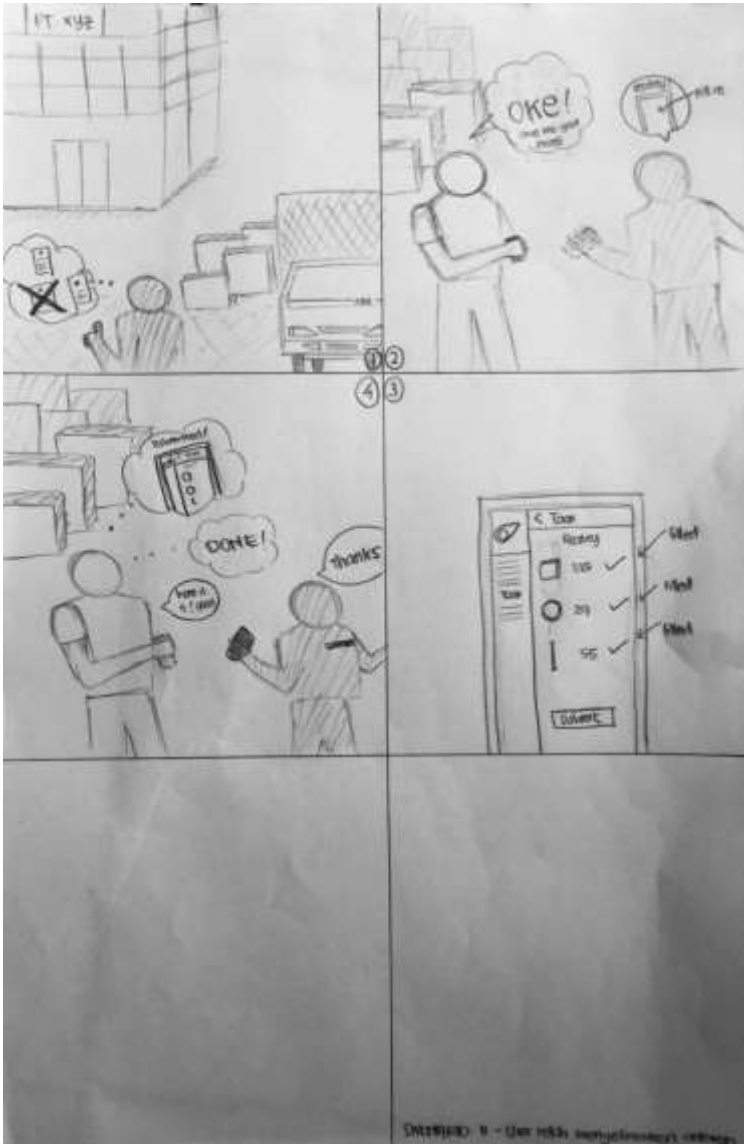
Lampiran 18



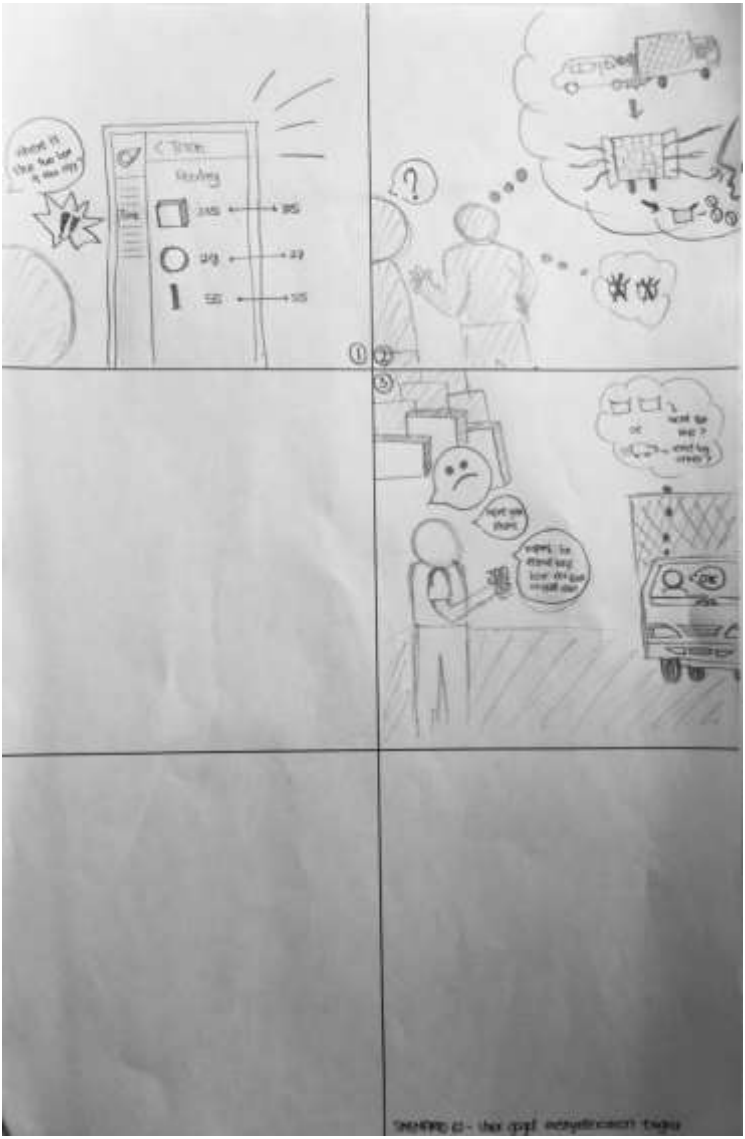
Lampiran 19



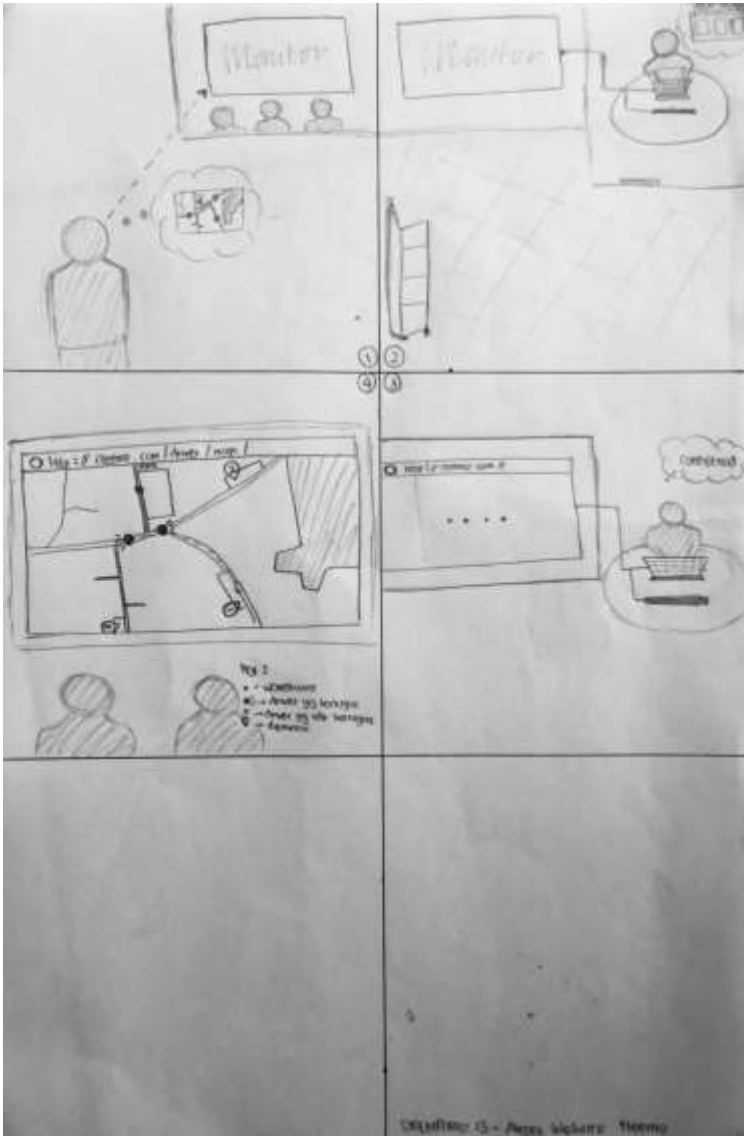
Lampiran 20



Lampiran 21



Lampiran 22



BIODATA PENULIS

Penulis yang berasal dari kota Pematangsiantar ini lahir pada tanggal 12 Juni 1995 dan merupakan keturunan asli dari orang tua yang berdarah Batak. Merupakan anak pertama dari 3 bersaudara, dan telah menempuh pendidikan formal mulai dari TK Cinta Rakyat Santa Lusia Pematangsiantar selama 2 tahun sampai tahun 2000, kemudian dilanjutkan pendidikan di SD RK Cinta Rakyat II Pematangsiantar sampai tahun 2007, SMP RK Bintang Timur Pematangsiantar sampai tahun 2010, SMA RK Budi Mulia Pematangsiantar sampai tahun 2013, sampai kemudian melanjutkan pendidikan di Jurusan Sistem Informasi FTIf – Institut Teknologi Sepuluh Nopember (ITS) Surabaya berstatus mahasiswa dengan NRP 5213100139. Selama menjadi mahasiswa, penulis aktif dalam organisasi baik di lingkup ITS maupun di luar. Untuk ruang lingkup ITS, penulis aktif sebagai kru ITS TV, yakni organisasi yang memonopoli peliputan berita dalam bentuk video dalam lingkup ITS. Untuk ruang lingkup luar ITS, penulis aktif di gereja sebagai pengurus Pelayanan Ibadah Sore dan Guru Sekolah Minggu. Pada tahun keempat perkuliahan ini, penulis tertarik dalam pengembangan teknologi terkhusus di bidang *Android* yang sedang sangat dibutuhkan masyarakat sat ini. Untuk itu penulis mengambil bidang minat Laboratorium Infrastuktur dan Keamanan Teknologi Informasi (IKTI). Untuk informasi lebih jauh, penulis dapat dihubungi melalui *email* dengan alamat dumoli13@gmail.com atau dumoli13@live.com.